COMMUNICATION-AWARE PLANNING AID FOR SINGLE-OPERATOR MULTI-UAV TEAMS IN URBAN ENVIRONMENTS

A Dissertation
Presented to
The Academic Faculty

by

H. Claus Christmann

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Aerospace Engineering

Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology
August 2015

COMMUNICATION-AWARE PLANNING AID FOR SINGLE-OPERATOR MULTI-UAV TEAMS IN URBAN ENVIRONMENTS

Approved by:

Dr. Eric N. Johnson, Adviser
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Eric Feron
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Panagiotis Tsiotras
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Karen Feigh
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: Mai, 15, 2015

To my grandfather;

who taught me to always rotate a 2 H *pencil when attempting to draw*

a straight and evenly thick line and that attention to detail matters.

ACKNOWLEDGEMENTS

"Standing on the shoulders of giants" is a metaphor often used in the context of academic achievement. What the saying doesn't capture is the fact that it isn't only the giants who support the achievement, but also all the helping hands that supported the sometimes tedious ascend to those metaphorical giant shoulders.

The first and foremost supporter of my journey is unquestionably my adviser, Professor Eric N. Johnson. Without his continuous backing I couldn't have explored all those related issues I was interested in and without his guidance and support I often would have been lost in the nooks and crannies. I want to extend my gratitude to the other members of my committee, Professors Egerstedt, Feigh, Feron, and Tsiotras, who inspired me to always consider multiple perspectives and to never forget about how all the little puzzle pieces only together form a bigger picture.

I am very thankful for all the help, support, and opportunities to learn new things I was given by my colleagues, fellow lab mates, and friends in Atlanta. From the early days onwards there were especially Nimrod Rooz, Suresh Kannan, Adrian Koller, and Allen Wu who showed me the ropes, be it related to maths, coding, soldering hardware into submission, or getting around Atlanta in general. Jeong Hur taught me much about building things that fly and that design for manufacture is important, at the latest when creating ones contraptions in the machine shop. Henrik Christophersen introduced me to the rigors of implementing avionics (and how not all rules have to be followed when fixing things in the field) and William Fenwick took on the burden introducing me to the South; I enjoyed all the time spent learning with and from them.

I am most certainly forgetting way to many names, but very many other people also helped along the way. Alison Proctor, Alex Moodie, Yoko Watanabe, Manuh Dhingra,

Jonathan Muse, Mike Sobers, Phillip Jones, Nathan Alday, and Fritz Langford all have given me at least one big push. So have all my current lab mates, Toshinobu Watanabe, Stephen Haviland, Dmitry Bershadsky, Lee Whitcher, Gerardo De La Torre, Daniel Magree, Takuma Nakamura, John "Jack" Mooney, Yong Eun Yoon, and Kyuman Lee.

Additionally, I don't want to forget to thank all the staff of the Daniel Guggenheim School or Aerospace Engineering for always being there and keeping things moving. Among them are Vivian Robinson-O'Neal, Susan Jackson, Howard Simpson, Scott Moseley, Scott Elliot and Rebekah Trout. I especially owe a very big thank you to Professor Jechiel "Jeff" Jagoda, who went above and beyond his duty and always was available for advice and support, no matter the topic.

Lastly I would like thank my partner Eva Baumert and my family. Without them, none of this would have ever been possible. *Danke.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

σ	Standard deviation (statistics).
A	Arithmetic mean (statistics).
F	F-Distribution (statistics).
L_4	Latin square of order four.
N	Generic count variable (specified in context).
t_W	Welch's t-test (statistics).
i	Generic index variable (specified in context).
p	p-value (statistics).
AGL	Above Ground Level (Altitude).
ANOVA	Analysis of Variance.
API	Application Programming Interface.
C2	Command and Control.
COTS	Commercial Off-the-shelf.
DDS	Data Distribution Service.
DEM	Digital Elevation Model.
FADEC	Full Authority Digital Engine Control.
FOV	Field of View.
GIS	Geographic Information System.
GNC	Guidance, Navigation, and Control.
GUI	Graphical User Interface.
GUID	Globally Unique Identifier.
GUST	Georgia Tech UAV Simulation Tool.
HAI	Human-Automation Interaction.
HALE	High Altitude Long Endurance.
HITL	Hardware-in-the-loop.

HMI	Human-Machine Interface.
ISR	Intelligence, Surveillance, Reconnaissance.
KML	Keyhole Markup Language (a XML notation).
LOA	Level of Automation.
LOS	Line of Sight.
MALE	Medium Altitude Long Endurance.
MANET	Mobile Ad-hoc Network.
MVS	Multi-vehicle System.
NCO	Network-centric Operation.
NED	North-East-Down.
PIMPL	Pointer to Implementation.
PIO	Pilot-induced Oscillation.
RC	Radio Control.
RF	Radio Frequency.
ROV	Remotely Operated Vehicle.
RTLS	Return to Launch Site.
SA	Situation Awareness.
SHF	Super High Frequency (3 GHz to 30 GHz).
SITL	Software-in-the-loop.
SLAM	Simultaneous Localization and Mapping.
SNR	Signal-to-Noise Ratio.
STEM	(The academic disciplines of) Science, Technology, Engineering, and Mathematics.
SVS	Single-Vehicle System.
TLX	NASA Task Load Index.
UA	Unmanned Aircraft.
UAS	Unmanned Aircraft System.

UAV	Unmanned Aerial Vehicle.
UDP	User Datagram Protocol.
UHF	Ultra High Frequency (300 MHz to 3 GHz).
VLAN	Virtual LAN (Local Area Network).
XML	Extensible Markup Language.
XSD	XML Schema Definition.

GLOSSARY

bifurcation graph The bifurcation graph is created from the route graph contracting all edges connected to at least one vertex of degree two. The resulting graph does not contain vertices of degree two, p. 45.

blip A blip is the representation one Node maintains of another Node, borrowing from its meaning as a radar display indicator of a reflected signal. The plural form "blips" generally includes the ownship unless the context creates a juxtaposition of ownship and non-ownship blips, p. 85.

command and control (C2) The Department of Defense (DOD) Dictionary of Military and Associated Terms [56] defines C2 as "the exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission." Command and control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the accomplishment of the mission, p. 14.

communication graph (com graph) The communication graph represents the datalink connectedness of the MVS network. Vertices represent MVS Nodes and edges represent the ability of the connected Nodes to communicate with each other. The communication graph is not necessarily connected, p. 28.

MVS control station (Control Station) A Control Station (capitalized) is a MVS control station node, i.e. a (stationary) interface which gives a human operator access to the MVS network, p. 1.

Δ-disc A Δ-disc is the circular region around a Node in which communications are possible where the radius of the disc is given by the maximum range of the utilized link. Two Nodes can communicate with each other if both are in the Δ-disc of the other Node, p. 53.

face graph The face graph is a (Delaunay) dual of the route graph. Its vertices, representing the faces of the (cyclic) route graph, are placed at the centroids of the faces and its edges represent neighboring faces in the route graph. The face graph has the same topology as the obstacle graph and there exists a bijection between their vertices and edges, respectively, p. 42.

hop In a networking context, a hop describes the transmission of a packet from one node to a neighboring node. The notion hop can be used to quantify the number of necessary transmissions on a route or it can be used tGustUavo describe the distance of nodes in a network topography map. In the latter, one hop is equivalent to an edge in a connected graph, p. 29.

human-automation interaction (HAI) The complex of how humans interact with any kind of automation, this context especially the interaction with automata in the form of autonomous unmanned aircraft., p. 6.

human supervisory control (HSC) Sheridan provides in [78] the following definition: "In the strict sense, [human] supervisory control means that one or more human operators are intermittently programming and receiving information from a computer that itself closes an autonomous control loop through artificial effectors and sensors to the controlled process or task environment. In a less strict sense, [human] supervisory control means that one or more human operators are continually programming and receiving information from a computer that interconnects through artificial effectors and sensors to the controlled process or task environment", p. 13.

level of automation (LOA) Originally proposed in 1978 by Sheridan and Verplank in [79], the levels of automation describe the involvement of a human in *"man-computer decision making for a single elemental decisive step."* The proposed scale ranges from one, *"human does the whole job up to the point of turning it over to the computer to implement"*, to ten, *"computer does the whole job if it decides it should be done, and if so tells human, if it decides he should be told"*, p. 16.

MVS network (Network) A Network (capitalized) is a group of MVS nodes which operate together, either at the same task or under the control of a common MVS control station. References to the Network as beneficiary of a certain feature mean that all participating Nodes as well as the human user who utilizes the capabilities of the Network, would benefit, p. 61.

network-centric operation (NCO) The announced aim of NCO is not only to connect plenty of comparable data sources (i.e. UAS, Satellite, AWACS, JSTARS, conventional ISR), but also to build a shared representation among the network users by utilizing this data throughout the entire network. The result is an immense growth in available raw data to each individual user, way beyond the very limited processing capabilities of a single (human) decision maker, p. 13.

MVS node (Node) A Node (capitalized) is any participating entity in a MVS network, p. xi.

obstacle cell Obstacle cells are the faces of the route graph., p. 42.

obstacle graph The obstacle graph represents the neighboring situation of the obstacles in the environment. It is a dual of the route graph and as such a Delaunay triangulation of the obstacles, p. 42.

open systems interconnection model (OSI model) Wikipedia [94]: "The Open Systems Interconnection model is a conceptual model that characterizes and standardizes the internal functions of a communication system by partitioning it

into [seven] abstraction layers. [Application, presentation, session, transport, network, data link, and physical.] The model is a product of the Open Systems Interconnection project at the International Organization for Standardization (ISO), maintained by the identification ISO/IEC 7498-1", p. 48.

outer cell The outer cell is the face of the route graph that is delimited by the operational boundary (on the outside) and the perimeter of the union of all other obstacle cells (on the inside). The outer cell is hence not affiliated with an actual obstacle *per se*, but with the operational boundary. The outer cell is the only non-simple polygonal cell, p. 42.

ownship The ownship is the Node which could also be called the host to the referring item or process. In the context of software, for example, ownship refers to the Node that executes the particular piece of software under consideration, p. 43.

remote piloting Remote piloting is a mode of vehicle control in which the remote operator controls the vehicle via a replication of control inputs that an onboard operator would use during manual flight. For flying vehicles, this most often means a replication of an onboard pilot's stick or yoke at the remote control station. As with direct piloting, stick commands can have different effects, depending on the current mode of operation. Remote piloting is a very common control scheme for all kinds of flight simulators or video games involving flight simulation, p. 2.

remotely operated vehicle (ROV) is a general term referring to uninhabited vehicles, independently of whether they are airborne, surface, or (under-)water vehicles. ROV also does not differentiate between remotely piloted vehicles and fully autonomous vehicles, p. 13.

RF graph An RF graph is a range-limited visibility graph. The edges of the graph connect vertices that, in a given environemnt, have a clear line of sight between each other and whose distance is smaller than a given threshold, p. 126.

RF polygon An RF polygon is a Δ-disc limited by line of sight (LOS) constraints, p. 53.

route graph The route graph is the cyclic subset of the Voronoi graph of the environment and provides a network of preferred routes for autonomous motion, p. 39.

situation awareness (SA) Endsley provides in [41, 42] the following definition: "Situation awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future", p. 2.

tactical interface operator (TIO) In a single-vehicle–multi-operator UAS the operators can often be grouped in at least two categories, one being more related to the actual piloting of the vehicle, the other, which TIO refers to, capturing the operators of onboard payload or other mission supportive systems., p. 10.

unmanned aircraft (UA) The Departiment of Defense (DOD) Joint Publication 3-30 [57] defines unmanned aircraft as "an aircraft that does not carry a human operator and is capable of flight with or without human remote control", p. 1.

primary unmanned aircraft A primary UA is the unmanned aircraft currently directly controlled through the operator. Primary UA provide the main sensory information to the operator by means of a live first-person-video feed which allows the operator to get a high(er) level of situational awareness. Since a primary UA is primarily tele-operated, it is not bound to predefined waypoints or routes or operational altitudes. The term describes a role of an unmanned aircraft and not a particular vehicle, p. 4.

supportive unmanned aircraft Supportive UA are all active unmanned aircraft of a Network which are not the primary UA. Supportive UA are are mainly utilized to establish and maintain connectivity in the Network. The term describes a role of an unmanned aircraft and not a particular vehicle, p. 4.

unmanned aircraft system (UAS) The Departiment of Defense (DOD) Dictionary of Military and Associated Terms [56] defines a UAS as “[t]hat system whose components include the necessaryequipment, network, and personnel to control an unmanned aircraft.” For the pupose of this work, UAS are comprised of one or several unmanned aircraft, one or several control stations, and the necessary equipment and infrastructure to operate those. The term UAS is generally used to more holistically describe the complete system, including its hardware (including the payload sensors and effectors), software, personnel, and also the capabilities it provides through the interaction of all of those, p. x.

tactical UAS in this work describes all kinds of UAS which are man-portable and do not need any kind of fixed infrastructure to be operational. The precursor *tactical* stems from the relation to the tactical decision making process, which these UAS are assumed to be supporting primarily, p. 1.

MVS vehicle (Vehicle) A Vehicle (capitalized) is a moving MVS node, p. x.

SUMMARY

With the achievement of autonomous flight for small unmanned aircraft, currently ongoing research is expanding the capabilities of systems utilizing such vehicles for various tasks. This allows shifting the research focus from the individual systems to task execution benefits resulting from interaction and collaboration of several aircraft.

Given that some available high-fidelity simulations do not yet support multi-vehicle scenarios, the presented work introduces a framework which allows several individual single-vehicle simulations to be combined into a larger multi-vehicle scenario with little to no special requirements towards the single-vehicle systems. The created multi-vehicle system offers real-time software-in-the-loop simulations of swarms of vehicles across multiple hosts and enables a single operator to command and control a swarm of unmanned aircraft beyond line-of-sight in geometrically correct two-dimensional cluttered environments through a multi-hop network of data-relaying intermediaries.

This dissertation presents the main aspects of the developed system: the underlying software framework and application programming interface, the utilized inter- and intra-system communication architecture, the graphical user interface, and implemented algorithms and operator aid heuristics to support the management and placement of the vehicles. The effectiveness of the aid heuristics is validated through a human subject study which showed that the provided operator support systems significantly improve the operators' performance in a simulated first responder scenario.

The presented software is released under the Apache License 2.0 and, where non-open-source parts are used, software packages with free academic licenses have been chosen– resulting in a framework that is completely free for academic research.

CHAPTER I

INTRODUCTION

Currently, many of the smaller-scale unmanned aircraft systems (UAS) aim towards high mobility, lower cost, reduced operator personnel, and ease of operation. Resulting from these design goals, most UAS employ a Control Station operator who pilots a single unmanned aircraft by means of tele-operation, using a live first-person video feed. The operator is most often assisted by additional ground personnel as well as some sort of stability augmentation to simplify piloting. This UAS category has been spearheaded through military systems like AeroVironment's Raven, EMT's Aladin, or Adaptive Flight's Hornet Micro, pictured in Figure 1. Their relatively simple operations can directly aid tactical decision making through their capability to provide live surveillance of the vicinity surrounding the Control Station operator and the immediate environment.

In recent years however, the civilian market has created a large amount of commercial off-the-shelf (COTS) multirotor systems which can also be added into the category of tactical-scale UAS, making "drones" a prevalent new element in the radio control (RC) hobby, in newly created commercial operations, and the related guidance, navigation, and control (GNC) research. Like their military ancestors, these civilian systems can provide the operator with a first-person video view, extended range communication and video links, and GNC capabilities in advance of simple stability augmentation: waypoints, flight plans, and even autonomous operations to a certain degree are readily available "out of the box." The 3D Robotics Iris+, the DJI Inspire-1, or the Ascending Technologies Firefly, Figure 2, are prototypical examples of this new type of COTS tactical UAS.

Although control at a higher level of automation (LOA), e.g. the use of individually preprogrammed waypoints or whole paths or trajectories, is often possible in tactical UAS,

(a) AeroVironment Raven **(b)** EMT Aladin **(c)** AFI Hornet Micro

Figure 1: Small tactical-scale UAS are man-portable, relatively cheap, and mainly used for locally limited ISR tasks.(U.S. Army photo by Spc. Michael J. MacLeod/Released; dapd; AFI

remote operators control the unmanned aircraft most likely also directly via remote piloting, providing them with immediate feedback (via the onboard video stream) and allowing them to perform tasks such as obstacle detection and classification, collision avoidance, and path planning at the same time, comparable to the tasks of a pilot in a manned aircraft operating under visual flight rules. Having the (onboard) vantage point of first-person video, the remote pilot can reach a high level of situation awareness (SA) [77], most often not reachable through other means of third-person tele-operation.

However, using first-person-video as a main control aid also severely limits the number of vehicles in a collaboratively operating multi-vehicle UAS to essentially the number of remote operators/pilots involved. Paralleling the development in manned aviation, with an increase in UAS complexity came the need for more human operators and larger-scale military high altitude long endurance (HALE) and medium altitude long endurance (MALE) UAS, for example Northrop Grumman's Global Hawk or General Atomics' Predator, require a crew of several people to be operated. Reducing the operator-to-vehicle ratio from several-to-one to one-to-several is an ongoing effort.[25]

An important contributing factor to human performance is SA. In a UAS with a vehicle-to-operator ratio larger than one, full SA of all vehicles at all times will hardly be achieved by a single operator, if at all sustained during a challenging mission. Even conventional large-scale RC model aircraft are often operated by a crew of two, the RC pilot and a

(a) 3DR Iris+ **(b)** DJI Inspire-1 **(c)** AscTec Firefly

Figure 2: In recent years a wide variety of civil COTS systems have been made readily available for beginners, aspiring commercial users, and the research community. (Copyright © 2015, DJI, Ascending Technologies GmbH. Used with permission.)

spotter within earshot. Moving the area of operations from wide open skies into an urban or otherwise cluttered environment increases the workload on remote pilots through a large number of obstacles to detect and avoid, smaller spaces to maneuver in, and tighter constraints on vehicle position and attitude in order to position the aircraft or its payload sensors in a useful way–all of which only further complicates efforts to achieve complete SA at vehicle-to-operator ratios greater than one.

1.1 Single-operator-multi-vehicle Systems

The driving motivator behind the presented work was the desire to enable a single human operator to interact with a team of unmanned aircraft in a meaningful way. A hypothesis in this context was and is that when an unmanned aircraft is remotely piloted, a single operator can only focus on that single unmanned aircraft, for example to gather sensor data. The sensor collection aspect of the piloting is important as the related direct control activity related to using, adjusting, and evaluating the sensor data can additionally increase the operator workload above a baseline resulting from operating an unmanned aircraft via a more indirect method with a higher LOA. A comparison in general or commercial aviation would be to compare pilot workload during cruise flight and during landing, for example. The latter, paralleling to the data acquisition task with the unmanned aircraft sensor(s) in the hypothesis, is a much more challenging task.

With that hypothesis in mind, the role of every individual unmanned aircraft in a network or team that is controlled only by a single operator, can be described as either the (sole) primary unmanned aircraft, the unmanned aircraft that is directly remotely piloted by the operator, or as one of the (many) supportive unmanned aircrafts, i.e. those unmanned aircraft which are not the direct focus of operator attention and which can be controlled trough an interface with a higher LOA. Indeed, if the team is, for example, in the ingress stage of a larger mission, no unmanned aircraft needs to fill the role of the primary unmanned aircraft as the operator, just like a pilot in cruise flight, can potentially monitor the unmanned aircraft team during this time of lower workload at a higher level and the control abilities available through the supportive unmanned aircraft interfaces could be absolutely sufficient. However, for a data acquisition task that utilizes a specific sensor, the operator might chose one of the unmanned aircraft in the unmanned aircraft team carrying a suitable sensor and finely control sensor usage, which would promote the chosen unmanned aircraft into the role of primary unmanned aircraft. This reflects the idea that a primary unmanned aircraft can do "more," as the operator, at the expense of a higher workload, has more access to more capabilities at a lower LOA (direct joystick control); complementary, a supportive unmanned aircraft that operates at a higher LOA (drag and drop) has the benefit of causing a smaller operator workload, but at the cost of a diminished set of capabilities.

1.2 Motivating Scenario

The research presented in this dissertation aims at overcoming limitations in numbers of unmanned aircraft effectively usable by a single operator by providing a single-operator multi-vehicle framework that allows a one-to-one ratio of one Control Station operator to one tele-operated unmanned aircraft, the primary unmanned aircraft, whilst keeping the two connected in beyond-line-of-sight (LOS) operations via a network built by data-relaying unmanned aircraft, the supportive unmanned aircraft. To guide the development, a hypothetical use case for a single-operator multi-vehicle system is used as a metric to weigh

(a) A fire fighter using a portable Control Station, preparing an unmanned aircraft for an inspection task.

(b) A deployed unmanned aircraft during ingress to the incident, providing live video to the Control Station.

Figure 3: In the motivating use case for a single-operator multi-vehicle UAS, a dedicated UAS operator, embedded into regular first responder forces, is tasked to support the first responders' efforts by providing meaningful ISR data in real-time to the rest of the squad. (Images: Copyright © 2015, Prox Dynamics AS. Used with permission.)

options and to provide a somewhat realistic context.

In this scenario, conceptually depicted in Figure 3, first responders in larger urban areas are envisioned to have access to tactical UAS to support and coordinate their efforts.[16] In one version of the scenario, the emergency response system would maintain several hangars, strategically distributed throughout the metro area, which function as remote dispatch centers for unmanned aircraft. First responders are envisioned to have a dedicated embedded UAS operator partaking in all missions. The UAS operator would, once on site (or potentially already during ingress), contact the closest hangar, request the dispatch of unmanned aircraft, and utilize them to support the other team members through the provision of live intelligence, surveillance, and reconnaissance (ISR) data of the incident. Alternatively, given the size of currently available COTS tactical UAS, several unmanned aircraft would be transported to the incident site in the response vehicles of the first responders.

A driving idea behind this scenario is the design goal to foremost provide additional supporting information to the first responder team. As such, the operator should primarily be a first responder and only secondarily be a UAS operator as this allows the operator to

use the obtained SA to directly process the "raw" data incoming from the payload sensors and, using expert knowledge as a first responder, relay a "processed" assessment of the data to team members most likely to make use of it. This requires an intuitive, low-workload UAS interface as the operator might otherwise suffer from inattentional blindness [82], especially in cases where full SA has not yet been achieved.

1.3 Outline and Contribution Preview

The following chapters of this dissertation will present the multi-vehicle system (MVS) developed to support the above presented first responder scenario, beginning with a brief overview on the background of human-automation interaction (HAI) in Chapter 2. The remaining document is following a bottom-up approach, starting with the presentation of the design of the underlying software architecture (Chapter 3), continuing with the provided algorithms and the related graphical user interface (GUI) (Chapter 4), and ending with a validation of their effectiveness (Chapter 5).

Chapter 3 heavily focuses on the foundations of the presented system. Section 3.1 introduces the basic architecture which enables the use of readily-available COTS UAS as well as the distribution of executables across several host computers, both console based as well as graphical. Section 3.2 describes how a geometrically correct representation of the environment is used to support the scalability of the presented system up to complete metro areas, including beyond-LOS operations through the utilization of a simplified simulation of radio frequency (RF) propagation, outlined in Section 3.3. Leaving the "user"-focused considerations, Section 3.4 introduces how the used software application programming interface (API) enables both "users" as well as "software developers" to do research with the presented system. In a similarly more developer-focused tenor, Section 3.5 introduces the intrinsic state machine framework and how it is used in the context of the utilized object-oriented coding paradigm.

Chapter 4 emphasizes aspects of MVS more "tangible" to (research) users: Section 4.1

describes the operating system agnostic GUI, Section 4.2 explains the command interface which follows a conventional mouse-and-context-menu approach. The remaining part of the chapter, Section 4.3, is dedicated to the developed aid heuristics, providing recovery from lost link situations and proposing team configurations that support a single operator in the ISR task posed in the motivational scenario.

Chapter 5 validates the effectiveness of the aids through a human subject study in which 25 participants were tasked to complete various missions replicating the motivating first responder task.

Chapter 6 concludes the dissertation with a summary of the contributions and a combined review and outlook to potential follow-up efforts.

Starting with page 162, the appendices primarily contain forms, questionnaires, and the collected data of the human subject study in a minimally processed form.

CHAPTER II

BACKGROUND

The overall usability of an UAS, or any system for that matter, is most often determined through a multitude of effects. If a single-vehicle–single-operator UAS is broken down conceptually as in Figure 4, the interconnectedness of the human operator into the UAS and its influences on measurable effects (e.g. in terms of mission success or failure) become apparent as a major control loop runs through the human operator via the power to *command* the unmanned aircraft. Given that changing the airframe, the payload, and the (physical) user interface is an involved process, Figure 4 highlights an alternative through changing the avionics.[1] Altering the "modes" as well as the related avionics software could change the usability, potentially leading to an improved incorporation of a human operator, which in turn, as the human operator is part of the UAS, could essentially result in a higher performance of the overall system.

Realizing that software changes affecting the avionics' modes can be done independently of potentially costly hardware iterations, improving the avionics can allow for potential improvements in deployed production systems through a software update.[2] But by the same token this also stresses that the development of those software components should incorporate the eventual end user, the UAS operator, from the early development stages as the quality of the user integration has an equally important contribution to the overall system performance as the airframe or the payload.

[1]The term *avionics* here is meant to primarily mean the GNC algorithms and the related software that affects behavior or handling of the unmanned aircraft.

[2]Once the control and navigation parts of the unmanned aircraft GNC algorithms are implemented, altering the guidance–the part of GNC that most likely resembles the *behavior* of an unmanned aircraft–can be cheaper and easier than hardware changes; at least if software validation is allowed to happen through (flight) testing or automated systems.

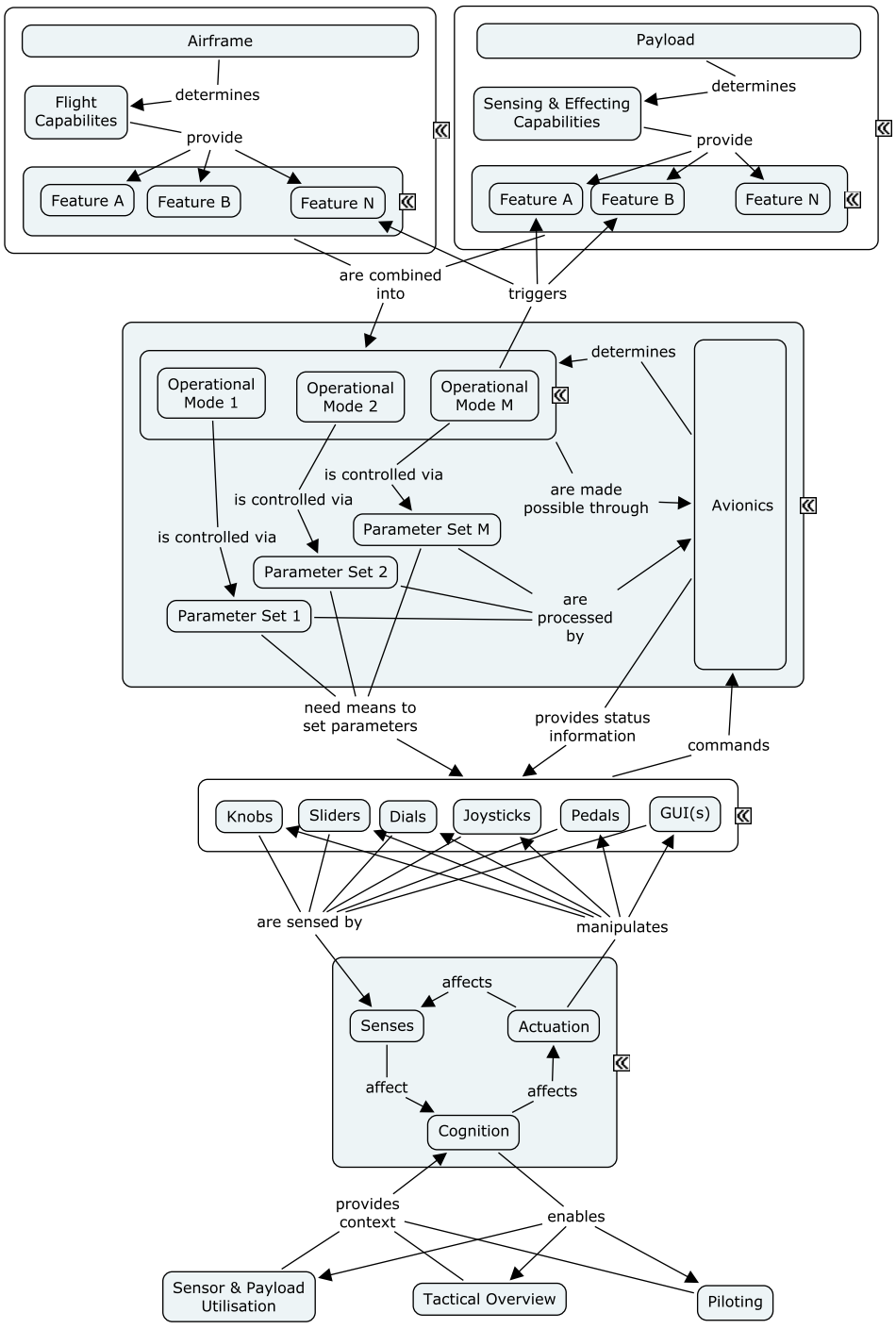


Figure 4: In a remote-piloting situation where the human operator is more commanding the autopilot than actually flying the vehicle, it becomes apparent that the design of the interface between the operator and the vehicle is deeply interconnected with the overall system performance. Both, ease-of-operation as well as the bad effects of mode confusion are situated within the design of those subsystems interfacing the avionics. Keeping existing physical constructs untouched (white groups), changes in the software of the avionics clearly can affect the HAI.

2.1 User Integration

As mentioned in Chapter 1, modern unmanned aircraft (like the tactical scale ones depicted in Figures 1 and 2, but also much larger HALE and MALE systems) are often tele-operated via remote piloting and not left in a completely autonomous state for prolonged periods of time. This underlines why a suitable design of the overall UAS needs to include looking at incorporating the human operator in a most supportive way. In an effort to better understand possible usability improvements through a higher level of adapted-automation-supporting HAI activities related to tactical aspects, an initial literature review started on aspects such as the workload of an tactical interface operator (TIO), currently deployed tactical interface technology, and how smarter automated path planning algorithms could support the reduction of TIO workload. In a single-vehicle–multi-operator setup, TIOs support the remote pilot through an interface sometimes referred to as the tactical[3] part or portion of the control station. This part mainly deals with flight plans, path planning and waypoint selection, selecting and activating single unmanned aircraft out of a unmanned aircraft team, tasking unmanned aircraft into different modes or activities, etc. The main question the literature review initially tried to find an answer to was: Which of their tasks would TIOs want to keep and which tasks would they happily hand of to automation?

However, after a review of the available literature it quickly became clear that a much broader approach to the introduction of HAI related arguments into UAS design is necessary in order to incorporate the user into a MVS tasked with an ISR mission comparable to the motivating first responder scenario. An overall awareness of the multi-faceted field of HAI in the context of UAS needed to be established in order to better design UAS (avionics software) for human operators.

Based upon the motivation of this work, consider the following scene in the scenario: a

[3]Here *tactical* refers to the military aspect of mission tactics. The earlier introduced term of a *tactical UAS* borrows from this as the main focus of UAS in this category is to support tactical decisions through the provision of appropriate timely local data.

team of first responders is tasked to deal with a fire in one of the upper floors in an urban mixed-use multi-story building. After the team has reached the scene, the team leader tasks two other team members with a simple task:

> "I need an overview. How can we get to the fire? Also, get an estimate on casualties, and figure out what is burning. The whole thing...
> Report back in 5 minutes. Go."

This scenario and the orders given could be assumed realistic and it seems that the tasking of the ISR mission is sufficiently detailed for all (humans) involved.

However, no currently available UAS is capable of replacing the first responders tasked with the ISR mission by a fully autonomous unmanned aircraft. Besides some other challenges (flying indoors in cluttered environments, the presence of fire, high temperatures, and low visibility) acting only on the given task description would be impossible. At the core the problem is the replacement of a human-human interaction (team leader with tasked members) with a human-automation interaction (team leader and UAS).

There are several problems associated with this replacement and for the purpose of this review on HAI it is assumed that physical and other engineering problems of any kind are solved or at least presumably solvable (like flying indoors, for example, or the actual physical sensing). What is still left are problems related to the replacement of capabilities in which even average humans far outperform even the best of automation: recognizing, interpreting, planning, adapting, and reacting. All of these activities are in domains where humans are far superior because they involve the incorporation of *context* into the data gathering (e.g. classifying information as pertinent or irrelevant) as well as the data processing (e.g. categorizing casualties based upon the severity of the injury).

As each of these capabilities–recognizing, interpreting, planning, adapting, and reacting–for the foreseeable future will involve human actors, each of these activities provides a different aspect for research on how to best integrate human operators into an UAS.

2.2 Human-Automation Interaction Aspects

Reviewing the literature on HAI for UAS systems, several different research foci become noticeable. There seems to be no common approach and, depending on the interest of the researchers and their projects, the interaction between humans and automation is broken down differently. Goodrich gives a broad survey in [48] and introduces various aspects that lead to breaking HAI down in these different ways.

As a result, this literature review is not able to give an exhaustive summary of the state of the art of HAI research in the UAS domain. Instead, this review aims at presenting a background on the different aspects researchers look at and tries to highlight the motivation for each–along with the resulting challenges and problems. Simple exposure to those aspects (and how and why they are important, which challenges they pose, and which benefits they promise) is a very important stepping stone for UAS designers not having a solid background in HAI or other cognitive engineering concepts as they can help to channel the use of the motivating first responder scenario towards more applicable questions. The findings of this review are a major motivator for the overall design guideline for the system architecture presented in this research proposal: keep it stupid simple.[4]

As an overall introduction, [25] gives a meta-analysis on different research topics. A more fundamental and much broader overview is given in [77]. The later also immediately presents the need to break HAI for UAS down into different categories or aspects as the article collection presents several headings under which the articles are sorted. Not necessarily matching the HAI aspects this work is approaching, the chosen categorization in [77] is as follows:

Human Factor Issues mainly deal with general issues involving humans and automation acting in concert.

[4]The KISS-principle (keep it stupid simple) seems to be in line with the goal of creating a shared mental model between the automation and the human operators. A too complex mode logic can lead to mode confusion (*What is it doing?*, *Why is it doing it?*, *How can I change it?*) even if the number of modes is limited, but an overly simplified system can hide important features.

Errors, Mishaps, and Accidents focuses on the deficiencies of humans (and the systems supporting them in overcoming those), resulting in "lessons learned'.

The ROV Interface is focused on direct interface issues: how to represent the automation to the operator and how to translate the operators intent into commands to the automation of the remotely operated vehicle (ROV).

Control of Multiple ROVs mainly deals with challenges resulting from leaving the current on-to-many ratio of vehicle-to-operators and the resulting loss of human support for low(est)-level decision making.

Team Control of ROVs instead focuses on how human teams successfully operate complete many-to-many systems in a coordinated fashion and what lessons could be learned from that for future introduction of (more) automation.

An essential concept that comes up in nearly every source reviewed is the concept of human supervisory control (HSC) (see [78] for details) and network-centric operation (NCO) . As a current trend in research aims at inverting the current vehicle-to-operator ratio towards several vehicles per operator, the core motivator behind HSC is the realization that the tremendous increase in raw data available to the operator results in the need for transforming the role of the human from a sensor-and-actuator into a supervising entity. Modern air transportation faces a similar issue, as Figure 5 hints at.

One aim of NCO is to not only connect plenty of comparable data sources, but also to build a shared representation by utilizing this data throughout the network. The result is an immense growth in available raw data way beyond the very limited processing capabilities of a single human decision maker.

2.2.1 Structured Representations

As a result of this supervisory approach to managing more data from more sources, one aspect of UAS related HAI research is focused on the (hierarchical) structure this interaction

(a) The Wright Flyer's Cockpit **(b)** Airbus A380's Cockpit

Figure 5: To realize why HSC is necessary, a simple comparison of the raw sensor data available to a pilot of the historic Wright Flyer to the data available to the two pilots of a modern commercial airliner hints to enormous data processing challenges resulting from more and more available raw data–even in or from only a single node in a much larger NCO, in this case the national airspace system. (Copyright © 2011 The 456th FIS Alumni Organization, Airbus. Used with permission.)

could be represented in. Inter- and intra-connections between data sources require data to be fused at some (central) point in NCO. However, centralization is prone to inject several layers in between the data source (e.g. an unmanned aircraft) and the actual user (e.g. a platoon leader or first responder).

[80] and [35] elaborate on a concept of how to structure this data path from the unmanned aircraft to a centralized node. A main focus of this aspect is the formation of regional clusters or regional commands. As such, research is interested in how to group unmanned aircraft, control stations, pilots, and intermediaries in order to reduce unnecessary hierarchies. This aspect ignores details on control station operations, pilot training, or sensor data evaluation and plainly focuses on a supervisory command and control (C2) structure in order to keep track of resources, cross-correlations and inter-dependencies.

2.2.2 (Human) Team Intra-Action and Team Structure

Complementing the previously described aspect is research focused on team structure and intra-team communication. [53, 54, 68] perfectly bridge from higher level HAI structures

down to questions related to team design. Focusing on a human team tasked with HSC of a limited number of unmanned aircraft (and hence forming one part of a multi-vehicle–multi-operator UAS), the spotlight is on choices related to the task allocation among the different members of the operator team. Design variables in this aspect are the team size, the breakdown of work related to supervising the UAS, and choices whether task allocation is static, dynamic, or hybrid. One focus of this aspect is the workload of each team member as a result of the team organization. Attention is also payed to interaction requirements of the UAS supervisory team to other levels of the overall C2 architecture.

Another part of this aspect, though slightly different in its focus, is research on the human team itself. Research is interested in what makes certain teams perform better in a given structure than other teams. [22] and [40], exemplary, present insight into the inner workings of human operator teams. A key concept is the establishment, representation, and utilization of a *shared mental model* among the members of a team. This is of particular interest as the articles show that teams perform better if they have this shared mental model. This model includes aspects of the members (i.e. if team member A announces X, then team member B takes this as a trigger to respond by providing Y), as well as the demands of the different positions (in a static task assignments) or roles (in a dynamic task assignment). The latter then aids the team members in better predicting and/or interpreting actions of other team members, all of which supports the overall team performance and SA.

Research on shared mental models (and particularly the related concept of *predictability*) is particularly interesting as a better understanding of them could lead to guidelines on how to replace or support human team members (or just particular roles of them in a dynamic allocation) with automation. (Section 2.2.4 elaborates more on levels of augmentation.)

2.2.3 Performance Metrics

Investigations related to this aspect are focused on establishing metrics which are objective and comparable. This is not only necessary to compare different approaches, but even more relevant for performance prediction and evaluation.

[30] presents a general taxonomy which classifies the different types of metrics usable across UAS. In [29] and [32] it is shown that a slightly adapted version of the Yerkes-Dodson inverted U-relationship (see [95] for details) also holds for multiple-vehicle–single-operator UAS.

A very detailed description of workload related operator performance for UAS is given in [39, 61, 92]. However, it could be argued that the presented experimental setup might not be representative for future UAS of this kind as the main (and measured) task requires a relatively low-level activity related to guiding single unmanned aircraft in the overall UAS setup.

[26] eliminates this aspect by tasking the operator in accordance to a HSC setup with purely supervisory activities by increasing the LOA of the unmanned aircraft in the study. Though not as detailed, the study presents an upper bound for operator utilization[5] of around 70 % after which the performance significantly drops.

2.2.4 Adaptive Automation and Decision Support

Unlike altering the scenario and/or distractions for human operators, this aspect keeps the scenario as a constant and introduced the level of automation (LOA) (originally proposed in [79]) as a variable whose effects are to be studied. [32] and [25] essentially study the effects of placing the human operator somewhere on the continuum from a (remote) pilot (i.e. a lower LOA) all the way up to something comparable to an air traffic controller (i.e. a higher LOA). The research focuses on how the task of the operator changes from level to

[5]The cited reference defines utilization as *percent busy time*. In this context this could be seen as the opposite to the concept of idle time of, for example, a CPU.

level and which tool provides most useful in accomplishing the mission. The main lesson learned is the restating of the nearly obvious: the higher the level of automation, the higher the level of abstraction necessary, and the higher the number of potentially supervisable unmanned aircraft.

Based upon the upper bound for operator utilization (see Section 2.2.3), the author of [26, 32] introduces automation to support a real time prediction of the workload throughout the rest of a simulated UAS mission. The proposed interface allows the operator to not only make decision based upon classically mission critical elements (such as ingress and egress routes, for example), but also based upon HAI aspects, for example, the expected workload during highly critical phases of the mission. Equipped with this tool, an operator could, for example, reschedule loiter and/or holding phases in unmanned aircraft specific mission plans such that high workload phases, e.g. target identification or engagements, happen in sequence and not in parallel.

In [28] the current aspect of adaptive automation is coupled with the previously introduced aspect of metrics and operator performance (Section 2.2.3). The work presents a mathematical model for coupling several types of *wait time*[6] with a measure capturing the LOA of the UAS. Based upon this, the model gives an upper bound on the number of unmanned aircraft a single operator could supervise at an acceptable performance and workload level.

In comparison, [89] and [11] slightly shift the focus, though are still concerned with LOA. They extend the concept of LOA into two dimensions, "Routing Task LOA" and "Allocation Task LOA." Based upon this, they investigate how independently adapting the two LOA dimensions based upon operator workload influences the overall UAS performance.

This last facet of the current aspect also introduces the notion of decision support systems (DSS) into the context of adaptive automation. At highly supervisory LOA levels,

[6] *Wait time* is comparable to the previously introduced concept of*idle time* and their counterpart *utilization*.

the SA of the human operator for the exact and detailed specifics of a particular unmanned aircraft at a particular point during mission execution can only be described as "vague" at best. In the case of unforeseen events that require an (potentially immediate) reaction or decision, there simply could be not enough time to establish full SA for the operator. [11] and [89] study the effects of different types of DSS, focusing on "management-by-exception" and "management-by-consent." The presented research elaborates on the performance increases with higher autonomy, but also highlight that reduced operator SA quickly introduces automation bias and complacency.

2.2.5 Interface Design and Human Factors

The lasts aspect to be presented is related to what could be called *ergonomics* in the context of UAS. The initial chapters of [77] elaborate on all kinds of issues which match the term *ergonomics* in a more traditional sense: posture, screen and interface positioning, control station layout, *etc. pp.*

However, the presented aspect is not solemnly interested in this type of traditional human workspace studies, but aims at understanding the peculiarities of human factors in the context of UAS. *Human factors* in this aspect is to be understood as effects resulting from mainly the facts which distinguish a human from automation. These could be "disadvantages" like imperfect memory, processing delays, and loss of focus or attention, but also "advantages" like shape and pattern recognition, adaptive decision making, and tactical and strategical planning, i.e. traits that relate to the identified capabilities that differentiate a human-human team from a human-automation team: recognizing, interpreting, planning, adapting, and reacting

[12] presents an overview on alternatives to vision based HAI systems and elaborates on different assignment strategies for alarms and other triggers to tactile or auditory interfaces. Along an identical line of argument, [24] investigates differences in interface layouts for several LOA. A (re-quoted) quote from one of the participants of the study presented in

[24] describes the studied UAS designer's dilemma in it's essence:

> "[Interface 3 is] a relief from all the raw data of [interfaces 1 and 2], but you lose in information precision, for example, what targets are reached."

As a reference for a complete description of a design of an UAS interface, [91] describes the steps taken in their study from the given team structure to the implemented interfaces. This work nicely complements the results from studies presented in Section 2.2.2.

In [65, 66, 73], the authors outline a detailed interface to support different LOA in the C2 interface between the human operator and the unmanned aircraft. Comparable to Calhoun's adaptive levels of automation (Section 2.2.4), the proposed interface pre-defines unmanned aircraft actions and mission building blocks (labeled "plays") in a hierarchical *why–how* relationship throughout all LOA. This allows an operator to quickly interact with unmanned aircraft by "calling the plays" and, if necessary, redefining the generic building blocks at any level and/or step necessary.

Reviewing mishap and accident reports, a loss of SA of the operator is often to be blamed. [21] presents a study on utilizing a shared concept of relative descriptors (e.g., *this* house next to *that* city square, whilst pointing at a map) to communicate intentions. This research is a good example of how the initially mentioned shared mental model could be utilized to better build HAI interfaces. For example, [67] presents a method to generate routes through an environment where the locations of turns are anchored at relative descriptors such as "turn left after the bookstore."

[62] elaborates on interfaces for dynamically assigned operator roles, as mentioned in Section 2.2.2. The concept introduces the notion of "operator call centers" for essentially outsourcing low level tasks or detailed decision making during phases of high workload of the principal UAS operator. This work, in the context of workload scheduling [27] and the overall HAI structure presented in [80], provides a matching proposal for the problem of dealing with limited (human) resources.

2.3 Unmanned Aircraft Systems as Human-Automation Teams

In [48] Goodrich provides a very broad survey of HAI from a fairly general perspective. The article provides a good foundation for an engineer not having been exposed to the specific perspectives, questions, and issues of the field and can kindle awareness of how a traditional engineering approach might miss important factors related to including human operators from the start. In [45] the review is tailored to UAS applications and Goodrich and Cummings provide a more concise perspective on human factors for the next generation of UAS. Based upon a review of past multi-vehicle studies they tie the requirement for a higher LOA to putting a human operator into a supervisory control in order to manage the workload resulting from controlling multiple vehicles. Placing the human operator in an outer control loop the encapsulates a traditional GNC inner loop and a loop they label *Navigation*, they differentiate the LOA for each of these loops. As expected, the inner loops always show the highest LOA, replicating that current UAS operator do not need to "fly" the unmanned aircraft but merely need to indicate their intentions to the (inner-loop) autopilot: for the majority of all use cases modern-day UAS operators do not need to worry about flight dynamics or stable flight conditions. The outer loops (*Navigation* and *Mission Management*) show lower LOA, reflecting the required increased input of the human operator. But while Goodrich and Cummings describe how the HSC modes can be managed in various ways, they also highlight a need for a more direct interaction to directly process data gathered by the unmanned aircraft, for example in a search and rescue scenario. In this context they also describe how the interfaces face different requirements for these two extremes of human involvement (direct "piloting" vs. higher level HSC) and how particularly in the directer control modes subtle changes in, for example, the orientation and inclusion of a video stream can support or hinder the correct correlation of the gathered unmanned aircraft sensor data in a broader context. ([23] elaborates a little more on the details and shows how payload operators, in the absence of kinesthetic cues and complete insight into the commands issued to a vehicle, did not realize that a newly

discovered target had indeed been previously found.) However, [45] does not seem to be totally applicable to the motivating first-responder scenario of this dissertation as all the investigated scenarios seem to always include a more global perspective on the use and usefulness of the unmanned aircraft-collected data. The studied cases apparently all had to deal with at least a two person UAS crew which in itself requires that the utilized systems allows those two operators to create a shared mental model of what is happening, which, as [23] shows, can not assumed to be given automatically. For the single expert-operator setup motivating this dissertation, these friction losses do not occur and the smaller or more local footprint of the scenario allows the expert-operator to be a sole collector and evaluator of the raw unmanned aircraft-data. However, this can easily change if the ISR character of the mission is replaced by a supportive one, for example when the UAS would be used to support several first-responder crews already engaging the fire. Such a follow up scenario would immediately introduce aspects of scarcity and the related task prioritization (the unmanned aircraft cannot support multiple request for data from different crews at different locations at the same time), which in turn could draw from the related work of Cummings on workload in a HSC context mentioned previously in this chapter.

In this context of integrating the UAS into an overall larger scenario, [46] present some practical experience of using UAS in a search and rescue operation. A main focus seems to be the approach taken to solve the (UAS) resource allocation problem and an optimized integration of the UAS capabilities in the larger effort. Again, in the motivational scenario for this dissertation such problems are not as prevalent as the described mission is of a much smaller nature. However, if the scenario were to be expanded, [47] presents some insights from a larger human subject study on how operators switch from a more direct or sequential control paradigm to a "playbook" style control (see [65, 66]) in order to manage the capabilities of the commandeered automation with the available or required autonomy level and information support systems.

Introducing UAS into larger scenarios, i.e. more unmanned aircraft, more operators,

more stakeholders, or more tasks or targets, the focus for the involved operators shifts towards a more supervisory role. As mentioned earlier, this is possible due to the advanced in the level of automation of the inner loops, relieving operators from "flying the aircraft", commanding an autopilot on what they want to happen. Based upon previous work on HSC (see Section 2.2), Cummings *et al.* in [31] present an approach to shift the operator focus from the vehicle to the task (of identifying targets). A main research interest in this realm is how a single operator can maintain an overview about the potentially complementary tasks and the relation to the vehicles involved in them. An important part of this are attempts to model HSC to better understand and analyze the effects of interface choices, [26, 28, 33].

An important related aspect to HSC are the different interface requirements. In [45] Goodrich and Cummings point out how direct control benefits from a first-person "chase" view, whereas supervisory control and data evaluation by non-pilots can benefit from mixed reality interface that overlay live (video) data onto a synthetic scenario to add (geographic) context. As mentioned before, [55, 83] flip this relation and overlay synthetic data onto the live video feed, creating an interface that is tailored more towards a "piloting" operator.

Adams *et al.* present a cognitive task analysis for the use of UAS in an wilderness search and rescue setting to determine how the data collected by an UAS can support the overall search effort.[5] A key aspect is the realization that the UAS is a limited resource (it cannot search everywhere at the same time), so some expert-opinion-based prioritization affects the deployment and the use of the collected data. Unlike in the motivational first responder scenario, the wilderness search scenario is a much more complex effort and the task analysis highlights even more how UAS can be used as "just another tool" that can be utilized best if the capabilities and limitations are known and the collected data is presented in a way that supports the evaluation through domain experts.

Related to this teaming, [2] elaborates how insights about SA in human-human teams could be translated into human-automation teams. The presented work combines SA levels

with methods of information processing an puts that in context of LOA, generally supporting that higher automation increases SA of the automated team members while decreasing SA of the human members. An interesting aspect is the notion of how the individual SA and a shared SA can be different in a team and that shared SA has to be build from exchanged information and a common understanding of how such data is interpreted. Humans suffer from imperfect memory and information losses when people hand over responsibility, for example due to shift rotations. However, it can be argued that in a situation like the motivational scenario of this dissertation, all automated team members, i.e. the unmanned aircraft, do not suffer from human-like imperfect memory or information losses due as a result of using digital communication at a common level of abstraction. As such, the notion of a potential difference in the individual SA of any of the automated team members seems to be contradictable, arguing that any gain in SA made by an automated team member can be shared (communicated) with the other automated members without an information loss–a capability that is not available to the human members of the team. While this ensures that all automated members can draw from a shared model of the environment, it does not solve issues related to establishing a shared SA between the human operators and the automated team members.

In [3] Adams presents an approach on how to design a human-automation interface. Adams breaks HAI down into six aspects–user centered design practices, human decision-making, workload, vigilance, SA, human error– and introduces their affects on HAI. The work presented in this dissertation does not always directly evaluate Adams' aspects, but some still apply. Vigilance for example, keeping the operator engaged throughout the mission, can be assumed to be given, as the motivational scenario is of a short duration and does not require the expert-operator to be idle until an alert as the prime task for the operator–using expert knowledge to evaluate the video stream–by definition keeps the operator engaged. However, the problem of salient errors in the autonomous supportive unmanned aircraft is somewhat related to vigilance if the operator is expected to have to deal with

supportive unmanned aircraft-related errors. Beside the other factors, the concept of utilizing user centered design practices is certainly important for the work of this thesis as the stated goal of combining unmanned aircraft pilot and payload operator roles into a single expert-operator would greatly benefit from a user interface that is tailored to this, presumably following the insights gained through a task analysis. However, while a task analysis certainly provides some insight, it must be kept in mind that human-human first responder teams cannot do the task described in the motivational scenario and as such no replacement of a human-human team with a human-automation takes place, but rather an expansion of capabilities. A task analysis of a human-human team as such cannot fully capture the intended use case, but it could establish which types of information deployed first-responders could find useful if the UAS mission is expanded from the current initial assessment to a continuing support scenario. In this context [4] is also of high interest as it presents a goal-directed task analysis of a first responder scenario in a chemical, biological, radiological, nuclear, and explosive search and rescue framework. The work interconnects the required (shared) SA with the habit of the observed participants to use sequential command and control of the involved robots, i.e. the operator tele-operates one robot at a time, resulting in "idle time" of the non-controlled robots. Although the first-responder scenario presented in this dissertation also uses a sequential setup (the expert-operator can only directly control one unmanned aircraft at a time, the *primary unmanned aircraft*), a notion of "idle time" or "neglect time" for the amount of times the supportive unmanned aircraft are not directly controlled might be miss leading as once placed, the loitering vehicles perform their task (relaying data), despite not being actively used. (Indeed, [55] mentions a similar use where in their study participants "parked" robots at certain locations to use them as a visual aid or landmark in a search and exploration task. The robot, despite not being actively controlled, is as such still serving the intended purpose.)

An interesting aspect also mentioned in [55] is the use of a halo-like ring around the (simulated) first-person view to indicate the relative location of the other vehicles. The

work presented in this dissertation sidesteps the mentioned issue of operators having to otherwise switch attention between a more immersive first person video (the primary source of data to be evaluated by the expert operator) and an overview granting top-down map by not providing a first-person feed. As such, the control station architecture presented in this dissertation seems to be more in line with an often mentioned two-person operator crew (a "pilot" who could focus on the map view and a "payload operator" who can focus on the video stream), but it can be argued that this is an artifact of having opted to not investigate such an immersive first-person interface further. For example, [5, 55] state a need for such a two person team with [23] expressing the hope that this can be abandoned in the future. Given the development in autonomous capabilities since the publications of these articles, for this dissertation it is assumed that automation has matured to the necessary level,[7] but that care must be taken in developing the ergonomics of such an immersive combined interface. ([83] presents an interface that seems complimentary to the halo approach taken in [55], possibly indicating that information about the periphery of the currently controlled vehicle can be presented in the peripheral field of vision of the operator.)

Looking at HAI from the automation perspective, [34] characterizes controllability in a leader-follower setup, a common scheme in HSC of multiple unmanned aircraft and [38] presents an implementation of a leader-follower setup in combination with a decision support system to optimize thread avoidance along a given flight path. Commanding only the leader unmanned aircraft, the human operator can effectively control the complete set of the followers, too, reducing the need to interact with individual unmanned aircraft at the penalties resulting from the indirection. The presented optimizations can, depending on the current setup, propose to use a certain unmanned aircraft as the controlled leader, which can minimize these penalties.

[7] In the presented form, MVS utilized the automation capabilities of GUST and during the performed human subject study, participants without prior exposure to RC (model) flight successfully "piloted" a SITL simulation that matches the dynamics of a Yamaha R-Max helicopter.

2.4 Discussion

Overall, ongoing research on HAI proves highly active and highly diverse. Different researchers focus on different aspects of the problem in an effort to brake it down into manageable pieces. However, the enormous amount of inter- and intra-connections, dependencies, and cause-and-effect chains of design decisions render the problem unsolved today.

The presented aspects are one way of describing different steps of an essentially iterative process, the design of a HAI interface for an UAS. As the fundamental system block in a human-automation team, the human, is far from being completely understood from a cognitive engineering perspective (or any other perspective, for that matter), no existing engineering model is able to completely predict the overall system performance of a given UAS HAI architecture, making model-based optimization impossible. Other than reverting to trial and error, the research approaches taken heavily rely on making simplifying assumptions in areas not apparently important for the current focus.

Resulting from this stems the belief that was used to incorporate the motivating first responder scenario into the developed multi-vehicle system: an inherently "stupid simple" system–though potentially limited in specific capabilities–can achieve a better global performance than one in which either the human or the automation cannot exploit all capabilities of the other. In simple systems, a major part of internal "friction losses" (generated through one team member trying to understand and exploit the other) can be avoided by reducing mode confusion, limiting the complexity of autonomous behavior, and consequently improving the automation's predictability. The overall simplicity can also be argued to enable and simplify the creation of a shared mental model between the human and the autonomous behavior, also a predicate strongly related to overall team performance.

The presented aspects of HAI were used to guide design decisions during the development of the MVS, whenever a selection between options had to be made. Revisiting the motivating first responder scenario under the perspective of the various HAI aspects, the simplest option that had not been ruled out was chosen.

Several aspects of MVS introduced hereafter are directly affected by the findings of the discussed papers:

One human operator, one primary unmanned aircraft. Directly controlling a unmanned aircraft at a (perceived) low LOA causes a high workload at the operator, rendering the execution of other tasks much more complex. MVS defaults to the utilization of high LOA supportive unmanned aircraft, only allowing a single primary unmanned aircraft per Control Station/operator. ([33, 45, 48])

Relieving the operator from "piloting." Control Station operators can directly control the primary unmanned aircraft via a joystick interface, but do not need to "pilot" the unmanned aircraft in the sense of stabilizing the flight of the aircraft. Even under direct control, the inner GNC loops of the unmanned aircraft maintain stable flight, enabling control at video-game like simplicity. ([33, 45, 48])

Supervisory control of supportive unmanned aircraft. Unmanned aircraft not under direct (joystick) control can be commanded at a high level via a drag and drop interface, relying on automation to determine a predictable collision free route and execute the traversal. ([2, 31, 45, 48])

Acceptance of sequential control paradigms. Operators under high workload conditions tend to switch to a sequential control paradigm, ignoring unmanned aircraft for times. MVS supports this through hover-capable aircraft that don't move if not commanded otherwise and by providing a completely cyclic route network that always allows non-hover-capable aircraft to loiter by flying a loop. ([31])

Support for "playbook"-style commands. Although not directly implemented, the available context menus for unmanned aircraft and other elements of the environment support "playbook"-style commands. The presented coverage aid can be seen as a single complex "play" to solve the problem of creating network coverage around a

target. ([47])

Tolerating human error. In anticipation of oversights by the operator, MVS provides a lost link procedure to recover from inadvertent loss of communication to a Vehicle. ([3])

Visualizing connectedness. A graphical representation of the (simulated physical) communication connections between the unmanned aircraft in the network allow the operator to increase the robustness of the utilizes communication network and detect imminent disconnect events through an (ignorable) graphical alert. (Communication links are color coded, links close to the maximum range are red.) ([34])

Adaptive network-coverage visualization. In addition to the constantly shown com graph, MVS highlights the area connected to the Control Station. This allows the operator to ignore the com graph and still stay connected to the controlled vehicle by simply staying within the highlighted area. ([3, 55])

Visualizing the sensor footprints. The top-down map view of MVS displays the FOV of the first-person video camera to support placement of the video feed in the overall environment and reducing the "soda straw" effect. ([23, 45, 83])

CHAPTER III

SIMULATOR FRAMEWORK

There are many COTS UAS in existence, but most of them are designed as a singel-vehicle system (SVS) and cannot be combined into a MVS, mainly due to limitations stemming from the underlying identification and communication backends; combining vehicles from different "ecosystems" is even more complicated. The presented MVS simulation framework puts forth a way to circumvent incompatibility issues through the provision of SVS adapted wrappers which allows the use of a unified simulation backend.

In the same way in which the introduced first-responder scenario motivates the operational aspects of the framework, the work in a research environment motivates some choices in the underlying software design. The simulator should be expandable, support "users" as well as "software developers," and support collaboration across researchers with different access levels to source code. Additionally, the framework should run in real time, on current hardware, potentially onboard.

This chapter, which is more tailored towards the "software developer" aspect than the "user" aspect,[1] starts by describing the high-level architecture of the presented simulator in Section 3.1, as it is motivated b the combination of single-vehicle simulations into a team simulation. This is followed in Section 3.2 by a description of the second building block, geometrically-correct environment generation, and the third building block–multi-hop communications–in Section 3.3. Section 3.4 continues the description of the software by looking into the framework's API, coding practices, and API requirements to include a particular SVS as well as the utilized state machines in Section 3.5. The chapter concludes with a brief discussion in Section 3.6 and an outlook on possible expansions of the

[1] Chapter 4 provides a more "user"-centered description of the capabilities of MVS.

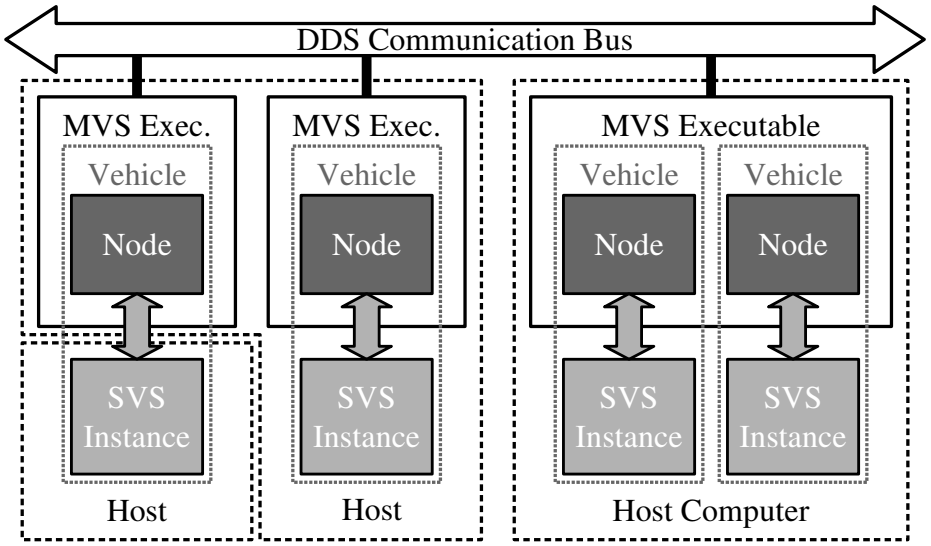


Figure 6: The communication architecture of the MVS framework which handles the global communication via DDS. The individual SVS instances do not need to communicate with or even be aware of each other as they maintain a one-to-one communication with a single MVS node.

framework in Section 3.7.

The abbreviation *MVS* will henceforth primarily refer to the presented multi-vehicle system, i.e. it will be used as a name for the presented framework. The third-party software utilized in the framework, as well as how to get access to the sources of MVS, are listed in Appendix A.

3.1 Software Architecture

One of the drivers for the development of the presented framework was the need to combine several individual SVS simulation instances into a larger MVS simulation. As such, the overall architecture of the simulation framework may best be described starting from a look at the desired inter- and intra-system communication setup, outlined in Figure 6.

Starting from a common one-to-one relation in the communication between the unmanned aircraft vehicle and its associated control station found in many SVS, the MVS framework uses the SVS's datalink as a fairly standardized way to establish an interface

with the SVS vehicle instance. Replicating the SVS control station instance, a Node translates the SVS-specific communication into a common protocol for use in the global MVS communication mechanism, thus translating an SVS instance into an MVS vehicle. This allows for a very minimal set of capabilities an SVS necessarily has to provide in order to be integrated into the MVS framework as a vehicle:

- On a GNC feature level, MVS only requires the SVS instances to support waypoint- (and possibly path- or trajectory-) following and the acceptance of some sort of direct user interface resulting from a manual control mode, e.g. through a classic remote piloting setup.

The only resulting datalink requirements are the messages related to transporting said information:

- On the uplink side, i.e. MVS to SVS, these are the direct control mode messages (e.g. from a joystick) and the waypoint, path, or trajectory information from a control station interface.

- On the downlink side, i.e. SVS to MVS, a status message with state information (position, attitude, and the related rates) and a message to track progress to the (previously uploaded) waypoint or along the path or trajectory are required.

This small set of datalink requirements limits the usable capabilities of the SVS instances' features, but it provides a smallest-common-denominator approach enabling the possible interaction amongst various (possibly heterogeneous) types of SVS instances in the MVS framework.

After the SVS-dependent datalink messages have been transformed at the Node level, MVS then uses a data distribution service (DDS) to establish and maintain the intra-system communication amongst all Nodes globally. The presented framework uses Real-Time

Innovations' Connext DDS Professional [74] as a COTS DDS solution provider.[2]

The use of the Connext system allows for an easy expansion of the MVS framework onto several hosting computing platforms, as the DDS-based intra-MVS communication automatically configures itself for various cases. MVS nodes can use the same methodology to communicate amongst each other, even if multipleNodes are run within a single executable (compare the two nodes on the same host computer in the right of Figure 6), if singleNodes are run in multiple executables and run on the same host (compare the twoNodes/executables in the left of Figure 6), or if Nodes are run in various configurations on various hosts on the same network. As the SVS datalinks normally are designed for inter-host communications (i.e. vehicle instance to control station instance), there is normally no need for an SVS instance to be executed on the same host computer as its associated Node (see far left in Figure 6). Combining all these options, the use of DDS allows the MVS framework to be distributed across several host computers which both enables the inclusion of particularly computationally expensive SVS instances as well as the collaboration of a large number of SVS vehicle instances; additionally, this setup allows for an easy expansion from the described software-in-the-loop (SITL) setup to a hardware-in-the-loop (HITL) test with actual (single) vehicles in a (possibly simulated) unmanned aircraft team.

3.2 Geometrically Correct Environments

Traditional approaches of representing a two-dimensional obstacle map as a simple free/occupied grid, representable, for example, as a black and white bitmap, can have benefits as the complete environment can be captured as a simple matrix or two dimensional array. The use of multiple vehicles, however, stipulates a use of the system in a large(r) environment, which results in the need for a large(r) map, which can quickly lead to performance problems as the map-underlying arrays grow quickly in proportion to the size of the covered area or the need for a fine(er) resolution. Common measures to avoid this situation range

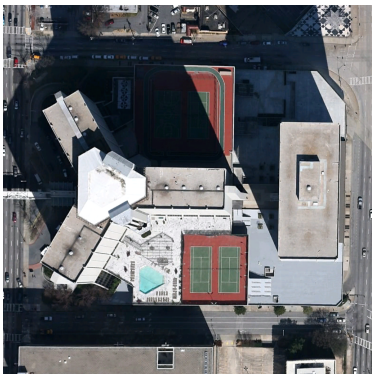
[2] RTI grants free academic usage licenses for their DDS Connext product through their University Program [75].

from simply limiting the resolution of the grid to using sparse matrix or quad-tree-based representations.

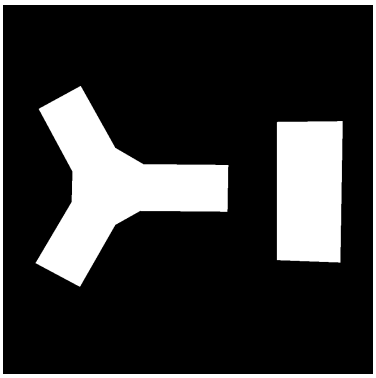
Figure 7 highlights such a grid-based approach: starting from an aerial photo, in this case a square picture with a resolution of 1024×1024 pixels, the environment is then segmented into free and occupied space (e.g. building/occupied as white and no building/free as black), resulting in a black-and-white map. This one-bit per grid element bitmap then needs to be stored and processed, which can happen in various forms: at the full original resolution (effectively Figure 7(b)), in some sort of smart or compressed way (Figure 7(c) presents a quad-tree based segmentation), or simply at a reduced resolution which trades memory requirements for mapping accuracy (Figure 7(d)–Figure 7(f)).

A potential problem with all these representations, however, is the inherent level of abstraction resulting from the trade-off of memory versus mapping accuracy, i.e. the reduction from more or less "realistic" obstacle data (be it from *a priori* knowledge or live from simultaneous localization and mapping (SLAM) techniques) to a set of grid elements that are marked as being occupied. By contrast, this abstraction is often what enables a multitude of algorithms to execute in real time, as the complexity of the underlying data can be greatly reduced.

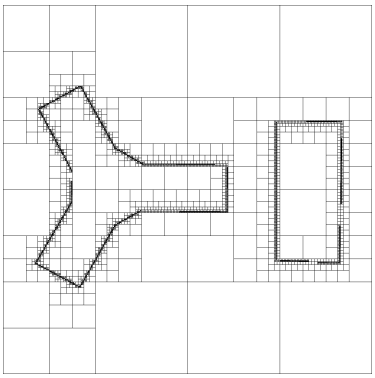
To counteract possible side effects from representing a potentially large map in any sort of resolution-compromised binary free/occupied grid, the presented MVS framework internally represents the free space in a two-dimensional environment via a geometrically-correct non-simple polygon. This representation does not need to sacrifice mapping accuracy for memory as the map is as accurate as the coordinates of the polygon's corners and memory requirements only correlate with the number of mapped obstacles and not the covered area. However, it does trade the availability of both large and accurate maps for the overall ease of code internal representation and processing of the obstacles, as instead of (code-wise) rather simple two-dimensional potentially sparse arrays, map-processing GNC algorithms now need to deal with the complexities of numerical geometry to determine



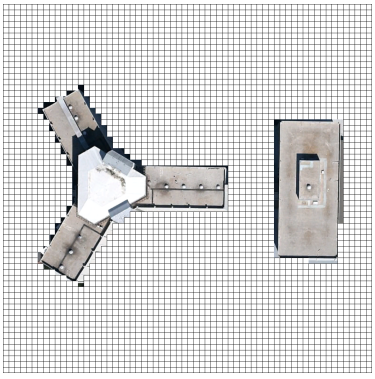
(a) Original 1024×1024 pixel image, 19 cm per pixel resolution.



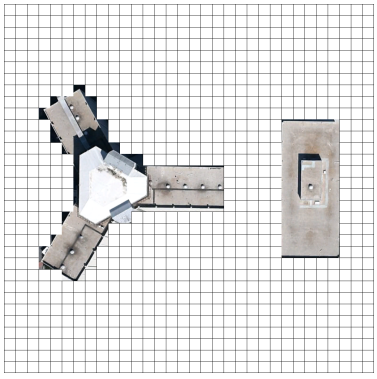
(b) Pixel-matching free/occupied grid, indicating the two buildings.



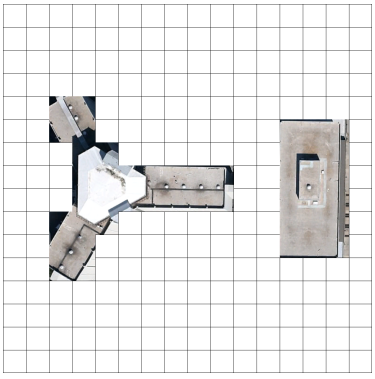
(c) Quad-tree decomposition of the free/occupied grid array.



(d) 64×64 grid, 16 image pixels per grid square, 3 m resolution.

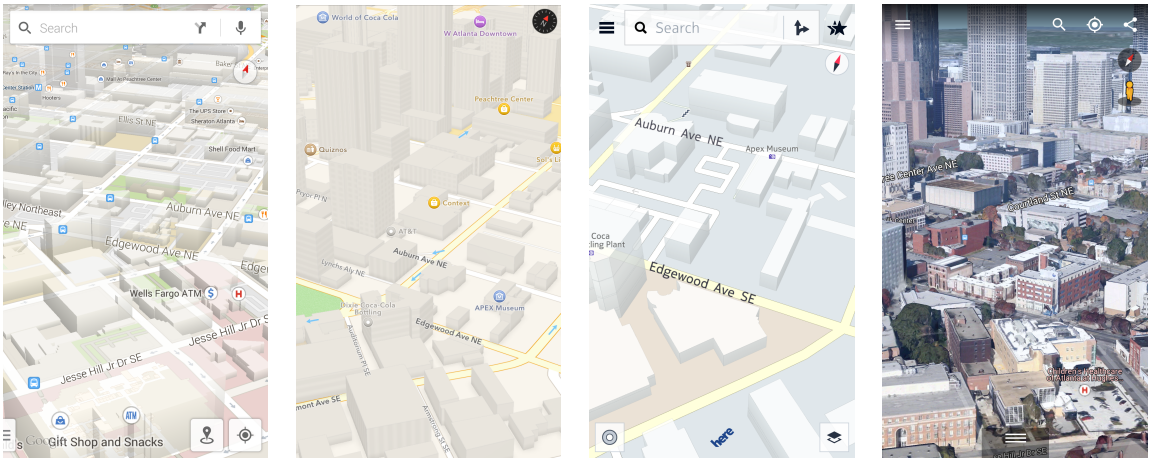


(e) 32×32 grid, 32 image pixels per grid square, 6 m resolution.



(f) 16×16 grid, 64 image pixels per grid square, 12 m resolution.

Figure 7: The use of grid-based free/occupied maps can lead to a need for large sparse map arrays and the need to use data compression techniques or a lowered resolution. (Image data: © 2010 Google.)



(a) Google Maps **(b)** Apple Maps **(c)** Nokia Here **(d)** Google Earth

Figure 8: Various freely-available applications for portable devices already have access to reasonably accurate three-dimensional obstacle data in larger urban areas. In certain places even indoor maps are publicly available through these channels. (Compare lower section of Google Maps screenshot, (a) and Figure 9. Copyright © 2015 Google, Apple, Nokia.)

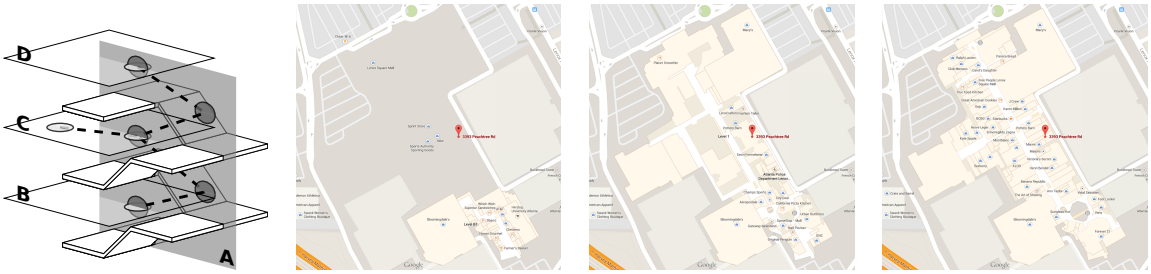
whether a certain area is indeed free to traverse or occupied by some sort of obstacle.[3]

3.2.1 Environment Generation

In order to generate this polygonal representation of the traversable free sapce, MVS makes a assumption: the presence of *a priori* map information available to MVS in the form of a Keyhole markup language (KML) obstacle file. The underlying rational for this assumption is the idea that digital elevation model (DEM) data and satellite imagery are widely available in a sufficiently high resolution to extract or generate an obstacle map using this information. Various geographic information system (GIS) software packages which allow the quick and easy overlay of two-dimensional polygonal data over such DEM-enhanced satellite imagery exist and presumably can result in a KML file to be used by MVS as a *de facto* free/occupied map of the environment.

A driving factor for the use of KML stems from the motivating first-responder scenario– if a map is not already available, the dedicated operator could use readily-available aerial imagery to generate a free/occupied map on the fly (in the first-responder scenario this

[3]MVS utilizes the Boost.Geometry library for these matters.[44]



(a) Stacked horizontal 2D maps can often be connected through a single vertical plane.

(b) A publicly available map of the basement floor of a shopping mall.

(c) A publicly available map of the 1^{st} floor of a shopping mall.

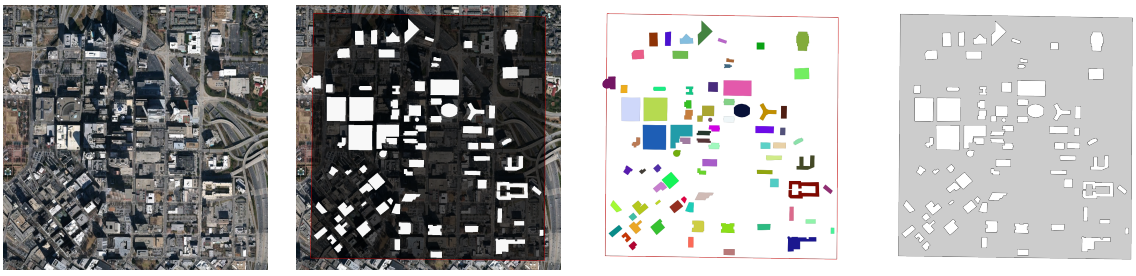
(d) A publicly available map of the 2^{nd} floor of a shopping mall.

Figure 9: Unlike natural structures, for example caves, most artificial structures tend to be "2.5-dimensional:" a two-dimensional (horizontal) floor plan extruded into the vertical. Urban environments mostly follow this pattern, especially in the interior of structures. The current limitation of MVS to two-dimensional maps hence still allows operation an a large set of structures, especially considering that interior maps start to become publicly available, too. (Map images: Copyright © 2015 Google)

could happen during ingress) by simply drawing on the georeferenced images using an overlay on any of the COTS software packages depicted in Figure 8.

A potential downside of this environment-generation scheme is the inherent limitation to two-dimensional environment maps. However, as Figure 9 shows, most man-made artificial structures tend to be more 2.5-dimensional: artificial structures, unlike caves, tend to have horizontal floor plans that are extruded "upward." As such, a stack of such two-dimensional maps can capture the significant features of many structures as the relevant changes in the third dimension happen in discrete steps. A single one two-dimensional map per "level" can capture nearly all of the relevant features.

The free-space map used by MVS is a combination of two things: an operational boundary and the obstacle data. Both are given to MVS in the form of KML files through the startup script (Section 3.4.3), and the launcher combines the boundary with the (covered) obstacles to a non-simple *Boost.Geometry* polygon by subtracting the obstacle polygons



(a) Satelite imagery of downtown Atlanta, Georgia, USA.

(b) KML-based obstacle data and boundary overlay in Google Earth.

(c) Imported items internally stored as simple polygons.

(d) Non-simple polygonal map; the interior represents free space.

Figure 10: Starting with widely-available satellite imagery, simple polygons can be used to indicate obstacles in the image through some sort of GIS interface. In combination with an outer boundary, the accessible (free) space can be represented as a non-simple polygon by subtracting the obstacle polygons from their encompassing boundary polygon. (Image data: © 2010 Google.)

from the boundary polygon.[4] This polygon then has an interior which represents the free space and an exterior which is given through both the area outside the boundary as well as the obstacles. The latter are represented as holes in the environmental polygon. Figure 10 outlines the conversion from a COTS aerial image to such an environmental polygon for a map of downtown Atlanta, Georgia. The generated map implies a certain operational altitude for which the map is valid. Not all buildings inside the area covered by the operational boundary have the same height, and for this reason the creator of the map has to decide upon an altitude level for which the map is valid.[5]

One drawback of the combination of an outer operational boundary and the inner obstacle polygons into a single non-simple environment polygon is the loss of identifiable obstacles. All obstacles are lumped together with the exterior of the non-simple polygon, which is why MVS does not discard the imported simple polygons representing the individual obstacles. The map portion of the MVS GUI, comparable to Figure 10(c), shows them as the basis for graphical representation.

[4]Treating obstacles intersecting with the boundary correctly still remains a TODO-item for the launcher code as the current method partially ignores them. (Compare the left boundary of Figure 10(c) and Figure 10(d).)

[5]Some of the software packages presented in Figure 8 support the drawing of three-dimensional graphics and could be used to provide an indicator for the map maker.

3.2.2 Environment Processing

Besides the shortcomings resulting from the lack of identifiable, user-clickable graphical objects, representing the environment as a single large non-simple polygon does not provide a good level of scalability for all types of algorithms used in a large environment. To maintain the high level of mapping accuracy available through the geometrically-correct representation while being able to function in the context of dynamic environments, MVS aims to retain and maintain as much information about the environment as possible which supports the provision of a set of "smart" features for environment processing algorithms such as pointers to the neighboring obstacles.

The underlying idea is that through the use of the geometrically-correct two-dimensional polygons, the complete environment is available at a very high resolution with the accuracy of the coordinates being used. It is not sensible to process the complete world environment algorithmically as a certain subset most likely suffices. Although results returned from such a run cannot claim to be "globally optimal" as available information has been actively truncated, they can be sufficient for many engineering applications. Examples can range from path-planning algorithms that focus on a neighborhood around a straight line between start and finish to highly-localized collision-avoidance methods which take only the immediate neighborhood into consideration.

MVS supports the creation and adaptation of the complete "world" environment to a localized subset through the provision of information on the neighbors of an obstacle, which is based upon a Delaunay triangulation of the obstacles. However, instead of computing the Delaunay graph directly from the obstacle data, MVS makes use of the fact that in $\mathbb{R}^2$ the Delaunay graph is a dual of the Voronoi diagram (of points[6]) and that a Voronoi diagram is computed already for guidance reasons (Section 3.2.2.1). Additionally, the duality-related

[6]The Voronoi/Delaunay duality is primarily given for pure point sites, but as the principal structure of a Voronoi diagram does not change when the points are replaced by small enough polygons, the (Delaunay) dual of the point-based Voronoi diagram is structurally identical to the one for the polygonal Voronoi diagram.

information is retained and provides a relation between the elements in the environment which can be used to enhance the performance of algorithms utilizing environmental data (Section 3.2.3).

In the current version of MVS, the processes described in the remainder of this section are only performed once[7] as the relevant results are directly related to the given static environment. Should the environment change, it is not always necessary to reprocess the complete environment, as the generated contextual information can be used to identify the parts of the world environment affected by a change. A possible future use case for this is presented in Section 3.4.4.

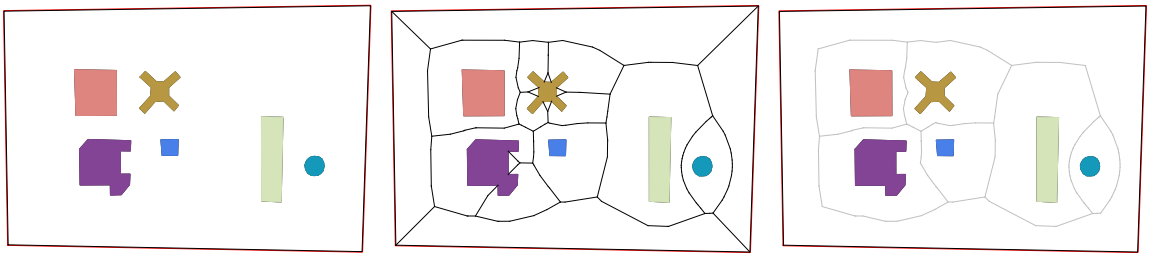
3.2.2.1 Voronoi-based Maximum Clearance Paths

A major benefit of having a map of an environment is the ability to plan collision-free paths and trajectories. Many different problem-solving methods have been proposed and research is ongoing in this field. For MVS, the presence of a geometrically-correct representation of the environment would allow for many other algorithms, and some could be more optimal from a purely shortest/fastest path perspective. The path-planning logic currently implemented in MVS aims to provide a scalable[8] way to compute a collision-free and easily-predictable path to guide autonomous vehicles through the mapped environment, as MVS uses Voronoi partitioning to indicate paths of maximum clearance through the environment.

The process of creating these maximum clearance route graphs is highlighted in Figure 11. MVS currently leverages the fact that a map is assumed to be available *a priori*, which is not strictly required as [58] gives a strategy to explore an unknown environment and build a Voronoi graph while doing so. MVS utilizes Held's VRONI [52] and the algorithm creating the Voronoi graph already provides additional useful intermediaries for free as they are

[7] MVS processes the initial environment during the initialization phase of a Node.

[8] The anticipated unmanned aircraft team sizes for a first-responder scenario are in the order to 10 to 20 vehicles and in urban envrionments of maybe a square-kilometer.



(a) The empty (training) environment. Only a simplified small version is shown.

(b) Voronoi partition; note the dead end branches reaching into concavities of the buildings.

(c) The route graph results from striping dead end branches from the Vornoi graph.

Figure 11: Starting with an empty environment, the route graph is created as the cyclic subset of the maximum clearance Voronoi graph for it. Eliminating dead end branches allows care-free use of non-hover capabale aircraft without fear of cul-de-sac problems.

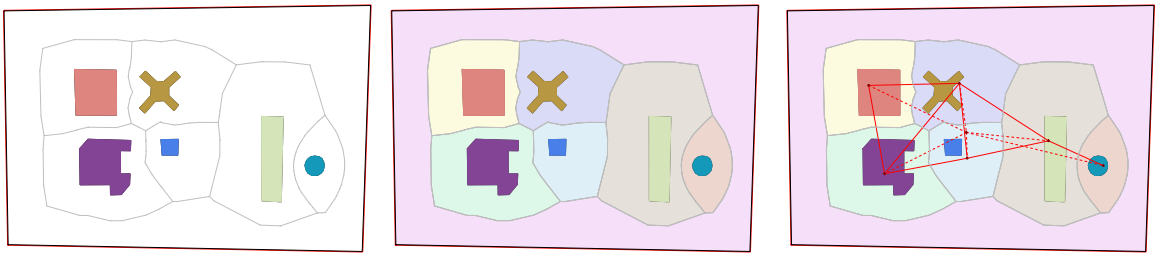
required for the originally sought Voronoi partitioning. For example, VRONI can provide offset curves around the obstacles at various distances which could be used in a possible extension of MVS to generate LOS-based shortest routes through the environment while ensuring a safe minimum distance from obstacles.[9] MVS limits the route graph to the cyclic subset of the Voronoi graph enabling the use of hover-incapable aircraft that have a minimum speed-dependent turning radius. With this restriction, the complete route graph is reachable for hover-capable and incapable aircraft. Because the route graph does not have dead ends, all aircraft can continuously stay in motion along its paths and avoid urban canyon or other *cul-de-sac*-related problems.[10]

3.2.2.2 Obstacle Neighborhoods

As mentioned previously, a dual of the Voronoi is the Delaunay graph. A node in the latter represents a face in the former and edges in the Delaunay graph indicate neighboring faces in the the Voronoi graph. As MVS provides a Voronoi partitioning of the complete environment, the neighboring data of obstacles can be extracted from that graph.

[9]LOS-based shortest routes certainly provide the shortest paths through an environment. However, Figure 43 highlights how safety margin considerations can lead to path solutions that are again closely related to Voronoi graphs.

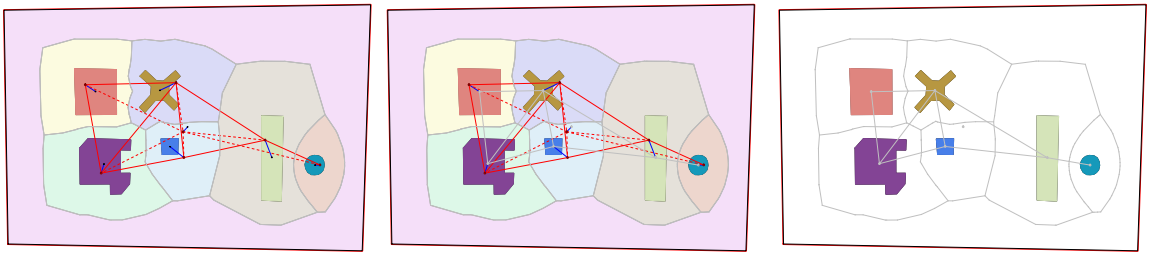
[10]The focus on the framework side was to not conceptually prevent hover incapable aircraft from being utilized in MVS. Some of the methods presented will not function with hover incapable aircraft without additional efforts.



(a) The Voronoi-based route graph is used to extract the Delaunay dual for the obstacle graph.

(b) The faces of the route graph form cells around the associated and contained obstacles.

(c) Using the face centroids as node anchors, edges in the face graph represent neighboring faces.



(d) The face centroid can uniquely be assigned to the centroids of the obstacles contained in a face.

(e) Replicating the face centroid edges with the obstacle nodes generates the obstacle graph.

(f) The resulting obstacle graph is a dual of the route graph, omitting the node for the external face.

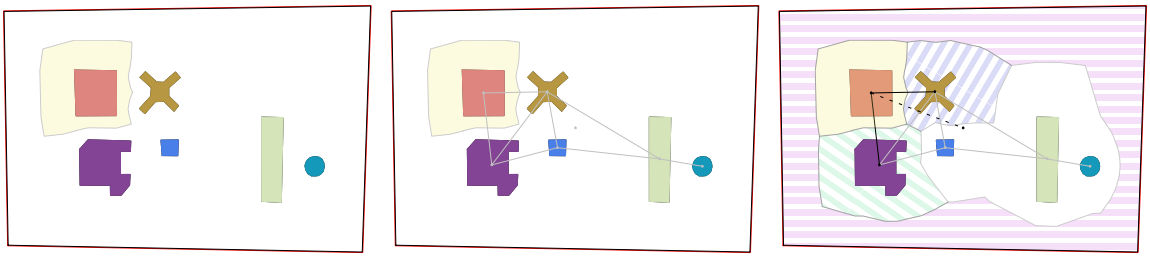
Figure 12: The obstacle graph represents the neighboring situation amongst the obstacles in the environment and is created from a dual of the route graph. Obstacles are neighbors if their associated face cells share an edge in the route graph. The representation of being a neighbor of the operational boundary, i.e. being an "outer" obstacle, is not represented in the graphics to reduce visual clutter, but it is maintained internally in code.

Although not the fastest way to get to the resulting obstacle graph, the process outlined in Figure 12 allows to maintain a connection of the obstacles to their surrounding free space and the edges in the route graph which circumnavigate the obstacle. These connections manifest in obstacle cells, the free space surrounding an obstacle for which a given obstacle is the closest. Figure 12(b) highlights the obstacles and the obstacle cells and depicts how the overall environemnt, delimited by the operational boundary, can be puzzled together from those obstacle cells, although allowing for a special outer cell that is not associated with an obstacle but with the operational boundary.

As the obstacle cells effectively represent free space, human MVS operators interacting with them could be confused as there presumably is no tangible item to pin properties too. To avoid this potential for confusion, MVS distinguishes between the face graph, a graph representing the neighborhood associations of the faces of the route graph, and the obstacle graph, a graph representing the neighborhood associations of obstacles. As faces and obstacles are mappable onto each other, both are topologically identical, yet the obstacle graph has the usability advantage that its nodes, representing the obstacles, are tangible–and hence clickable–items. Figure 12(e) shows the face graph in red, the obstacle graph in grey, and the related bijection as blue arrows. The figure also shows how the graphical representation of the obstacle graph omits edges which would connect to the obstacle graph node associated with the intangible and unclickable "obstacle" created through the operational boundary.

3.2.3 Environment Usage

As briefly mentioned during the introduction to Section 3.2.2, MVS aims at providing a code backend which can provide "smart" interactions with the underlying environmental database. The following examples showcase methods to make use of the relation between obstacles, cells, and the routes in order to better suit or improve algorithms which process the environment routinely.



(a) The route graph faces highlight the areas in the environment closest to a particular obstacle.

(b) The obstacle graph represents the possible traversals into other, neighboring cells.

(c) The cells at a one-hop distance are all the obstacle cells a node can possibly enter.

Figure 13: Based upon the knowledge of the current "closest obstacle," the overall environment can be segmented into three regions: the current cell, neighboring cells, and all others. This discrimination can be easily maintained whilst moving through the environment as once the current cell has been left, only (the old) neighboring cells have to be checked for potentially having been entered.

3.2.3.1 Contextual Environment Segmentation

As the route graph, basically being a Voronoi partitioning of the environment, segments the two-dimensional map-based upon the distance to the surrounding obstacles, the faces of the route graph, the obstacle cells mark the set of locations in the environment which have the same obstacle as their respective closest one. In the context of motion-planning, this has the benefit that if the current obstacle cell is known, the currently closest obstacle is also known through the bijective relation between the face graph and the obstacle graph. This information could, for example, be used by a collision-avoidance algorithm.[11] In the context of a HITL simulation, knowing the closest obstacle and its neighbors can also be used to implement a simple culling of the environment during the generation of simulated sensor data. For example, the algorithm generating the simulated sensor data does not need to sample the complete environment, but can limit its efforts to a region focusing on the current local neighborhood.

MVS utilizes an approach comparable to the latter example during some computations for a GNC algorithm in which a localized subset of the overall environment is sufficient

[11] Most likely a collision-avoidance algorithm would need to know more than just the ownship position as, depending on vehicle type, momentum, and velocity, a euclidean "closer" obstacle behind the vehicle could be of much less concern than a "farther" one directly in the path ahead.

(Section 4.3.2.2). Using the obstacle graph, the complete environment is segmented into three regions (compare Figure 13(c)): the current obstacle cell (solid), the immediate neighboring cells (striped), and all others (blank). As Figure 13 shows, this is a direct representation of the zero-, one-, and two-or-more-hop distance cells in the obstacle graph. (Normally the obstacle graph does not show the edges between the outer obstacles and the node representing the outer cell to reduce visual clutter. Figure 13(c) deviates from this and shows the relevant edge dashed.) To reduce the computational burden for that algorithm, MVS limits the size of the processed environment to the current and the directly neighboring cells, i.e. the zero- and one-hop distant cells, discarding any of the two-or-more-hop distant ones.

However, this method does have its own potential pitfalls once one of the included cells is the outer cell (the horizontally striped cell in Figure 13(c)), as then the holes present in the reduced environmental polygon might have to be interpreted differently. While some of the holes match with obstacles, there will be one hole that corresponds to all the omitted obstacle cells. From an algorithmic point of view, the boundary around that hole should be more comparable to the outer boundary of the polygon (representing the operational boundary) than to the boundary of an obstacle. The difference would occur, for example, in the generation of false data for a range-finding sensor: intersecting a virtual beam with an obstacle related hole should return the proper distance measurement to that point whereas an intersection with the outer boundary should result in a range reading of "infinity" as even if there were obstacles outside the operational boundary, their presence would have no effect as vehicles are not allowed to move there. Following that same logic, an intersection with the hole related to the omitted obstacle cells also should cause an infinity reading as, if the presence of that obstacle would matter, this should have not been omitted in the first place. The problem with this approach is that now the algorithm needs to not only differentiate between the inner and the outer boundaries of the free-space environmental polygon, but it would also need to know more about the holes, which would defeat the purpose of simplifying the environment into a single non-simple polygon in the first place.

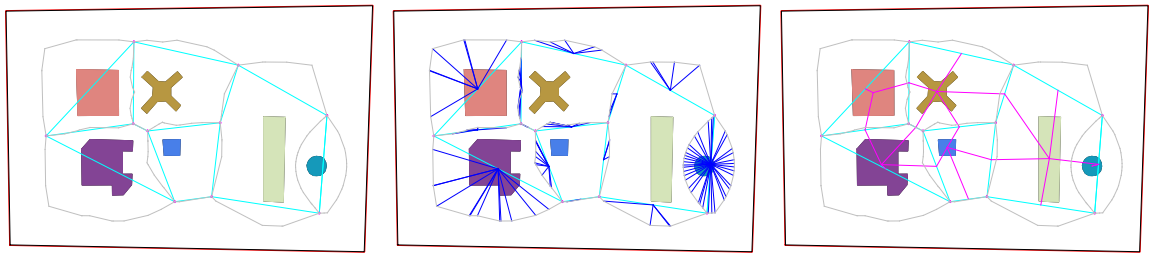
3.2.3.2 Contextual Routing and Path Planning

Keeping the contextual connection between the route graph and the obstacle graph can be used to speed up the processing of the environment, for example in cases where a human operator of a Control Station selects a specific obstacle as the target for an inspection task (Section 4.3.2.1). A fundamental task in such a scenario is the provision of a route to the just-indicated target.

While the internal representation of the traversable free space as a non-simple polygon should allow for a variety of planning algorithms, the presented implementation of MVS provides a routing mechanism which heavily utilizes the computed route graph. The use of these maximum clearance routes does indeed limit the possible paths through the environment to that given set of "railroad tracks in the sky," but this limitation increases predictability of the selectable paths to a human MVS control station operator. Additionally, staying on the computed maximum clearance paths allows for a fair amount of inaccuracy in the outlining of the obstacles as the paths automatically maximize the safety margin to all surrounding obstacles.

The reduction of the freely traversable (two-dimensional) free-space to a given network of paths simplifies the planning considerably as it is now comparable to simple routing commonly done for cars or other ground vehicles utilizing any sort of predetermined path network. On a very simplistic level, the routing can be reduced to a sequence of "turns," announced or processed whenever a vehicle reaches a fork in the path network.[12] The planning task can be simplified without loss of generality by simply probing a graph representing the actual bifurcation vertices of the path network in which the edge cost for the connection between two forks has been augmented with a cost representing the underlying multi-vertex path. In MVS, this graph is called bifurcation graph; Figure 14(a) depicts the relation between a bifurcation graph (cyan) and the underlying route graph (gray).

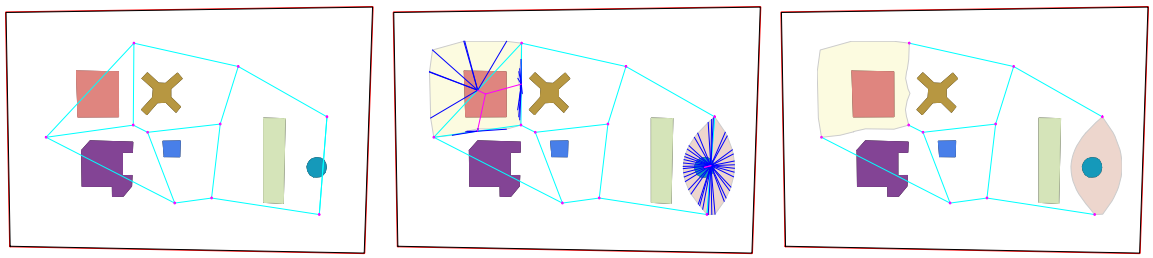
[12] This obviously ignores various complexities of multi-lane traffic and other common enhancements of the present day road network.



(a) Eliminating all degree-two vertices from the route graph yields the bifurcation graph.

(b) The bifurcation graph edges maintain an association with the eliminated vertices.

(c) The bifurcation graph edges also maintain a connection to their associated obstacles.



(d) The resulting bifurcation graph greatly reduces the graph complexity, yet maintains contextual details.

(e) Using the obstacles' contextual information, the routing network's complexity can be greatly reduced.

(f) Maintaining only a high level of detail in the source and target cells ensures planner/-ing scalability.

Figure 14: In a typical routing task from a source to a target cell, the contextual connection of the route and bifurcation graphs can be exploited to greatly reduce the complexity of the planning network whilst maintaining the highest level of detail where it is ultimately required.

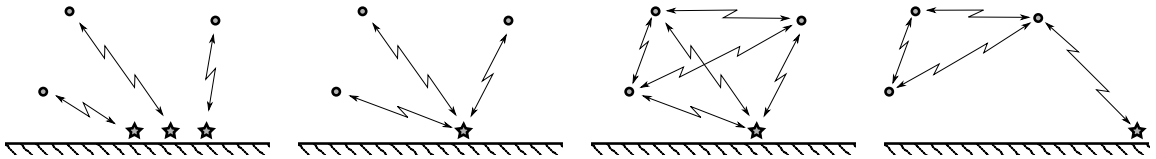
In the mentioned task example of routing a Vehicle to a certain target, the contextual information of the related obstacle cells can be used to generate a combination of route and bifurcation graph elements that maintains the high vertex resolution of the route graph only where it is necessary and replaces it with simplified bifurcation graph edges where not. Figure 14 depicts this process for an exemplary routing problem. The resulting graph (Figure 14(f)) is a combination of the route graph for the source/target cells and the bifurcation graph edges that connect them. One usage scenario of this setup could be the use of preplanned flight trajectories: the entry and exit points of the bifurcation graph edges are known, so a sequence of waypoints (possibly matching the original route graph vertices) can be precomputed and simply retrieved from memory when said edge is to be traversed.[13]

3.3 Multi-hop Communications

Other than large environments, another intrinsic factor to be considered is how to capture and replicate how the Node elements in MVS communicate in that environment. On a code-level, MVS opted to use a DDS messaging system, but that does not touch upon the RF related aspects of communicating in a structured environment.

Obviously SVS also include communication, but traditionally most systems do not focus on the various complexities and potentials that arise in the context of allowing more than two participants to partake in the communication network. A good example of this could be the maximum distance a vehicle could be send away from its control station. In an SVS, this is more or less given by the maximum range of the employed point-to-point link between the vehicle and the ground station. The range of a wireless link can roughly be estimated through a link-margin computation, which, after being confirmed in a field test, could then be set as a hard limit in the system, effectively creating an upper bound on the operational distance of the vehicle. If the range is not satisfactory, the only thing that can be done easily is to affect the dominant inputs in the link-margin formula: gain of the utilized

[13]This approach could be compared to an approach to control an unmanned aircraft through a sequence of "open loop" maneuver primitives which are stitched and smoothed to generate a final trajectory.



(a) An SVS setup requires three Control Stations to have three vehicles airborne as individual point-to-point links have to be established.

(b) Connecting the radios to a single Control Station creates a star-like RF topology with the Control Station as the sole exchange point.

(c) Enabling all participants to communicate with each other can increase complexity, but also the robustness.

(d) Using the inter-vehicle communication capability allows a topology changing increase of the covered area.

Figure 15: The communication network setup of an SVS is dominated through its point-to-point character. Although it is possible to create an MVS communication network by operating many point-to-point links in parallel, the easier setup is paid for by limited possibilities and the introduction of a bandwidth bottle neck for vehicle-to-vehicle traffic. At the cost of a more complex communication setup, enabling a full connectivity amongst all network participants eliminates this bottleneck and expands communication, possibly to a larger range.

antenna, output power, and frequency, all of which only affect the communication link on the physical layer of the OSI model. To a certain degree, changes in the layers above that are also possible, but to simplify all that can be done to increase the range is increasing the robustness of the utilized point-to-point link, essentially making the link work at a lower signal-to-noise ratio (SNR) at the the cost of a reduced bandwidth and the benefit of possibly squeezing a little more range out of the margins. In an MVS, all this is possible, as every individual link between two participating nodes can employ these tools as well. But unlike in an SVS, in an MVS the effective maximum range can nearly be doubled, simply by using one of the nodes as a relay.

What makes MVS communications fundamentally different from SVS communications is the potential to not only actively affect the topology of the communication network, but to also use the possible topologies to an advantage. Figure 15 depicts how the switch from three parallel utilized SVS to an MVS could be used to realize the above-mentioned range extension. A few related things immediately surface in the context of using and incorporating the network topology to an overall advantage for an MVS:

- Taking a given network topology, what can be done in the various layers to improve link performance in terms of throughput, delay, and overall scalability?
- Taking a given network abstraction, how can the topology be altered to improve the communication link in terms of robustness, range, and maximum coverage?
- How can the two factors above be used in combination?

In the context of a software framework that should be able to replicate all this in SITL as well as HITL setups, those points require that the framework essentially can simulate the communication network, including the lower layers of the OSI model as some desired benefits are only realizable there. Simulating networks is an ongoing research effort in various communities outside of aerospace engineering (e.g. [70]) and and can be investigated all the way to the simulation of RF propagation through a given environment (e.g. [69]).

3.3.1 Urban Environments

Looking at motivating scenarios for MVS, it appeared that two could especially benefit from the utilization of multiple vehicles: search operations over large areas and operations in urban environments. Performing search operations in a large area with multiple entities in itself is a challenging research area from the planning perspective;[14] from a communications point of view, the scenario is a suitable application for many types of mobile ad-hoc networks (MANETs). The urban scenario condenses this application as buildings quickly interfere with RF communications and multiple vehicles could be required to span a MANET due to loss of LOS much earlier than due to range limitations.

The utilization of unmanned aircraft in the context of urban operations to span and deploy MANETs is certainly a very interesting field, touching upon aspects ranging from infrastructure independence (for example, disaster response actions) to increasing communication robustness in potentially hostile RF environments. After some initial attempts

[14] Dataferrying and high latency MANETs are a closely related topic as it also includes loss-of-link scenarios and other restrictions which can occur due to the used hover incapable aircraft.

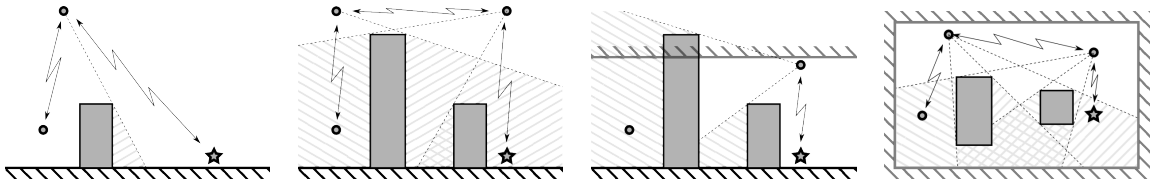
investigating MANETs [13, 17–19], the decision was made to *not* include a sophisticated simulation of the communication network environment into MVS, but to provide a suitable hook in the sources for other simulating software to tie into MVS to perform these steps.

However, even without a sophisticated simulation of RF propagation or the communication mechanisms across all seven layers of the OSI model, some interesting insights into operating unmanned aircraft in the challenging RF environment of modern urban areas can be obtained even with a very simple simulation. To do this, MVS lumps the challenging urban RF environment factors into some broad assumptions:

- Buildings drastically limit the RF propagation, i.e. they are RF opaque.
- Urban areas tend to have RF activities in the complete spectrum conventionally used for unmanned aircraft communications, i.e. the environment is very "RF-noisy."

As a result, MVS makes the assumption that the effective range of the utilized datalink is drastically reduced compared to the theoretically-achievable distances in a more "RF-quiet" environment. This assertion is based upon the idea that obstacles in the environment increase the attenuating effects and the presence of various third-party RF sources increases the general noise floor, decreasing SNR and also limiting the achievable distances. In addition to these range limitations, MVS also assumes that the utilized datalinks essentially only work within a direct LOS, i.e. not through buildings. This limitation stems from the observation that increased throughput requirements tend to be realized with higher frequency links (predominantly in ultra high frequency (UHF) and super high frequency (SHF) bands) which have limited diffraction, primarily propagate as direct waves, and get heavily attenuated by city structures. Depending on the utilized datalink, the presence of other third-party transmitters can furthermore limit the temporal availability of relevant spectrum segments which can result in reduced bandwidth and throughput.[15]

[15]This is comparable to the throughput enhancing 40 MHz channels in IEEE 802.11n. The standard specifies them for the 2.4 GHz as well as for the 5 GHz band , but due to the general congestion in the 2.4 GHz band



(a) Placing a single node high and behind obstacles in cases of few obstacles.

(b) Relaying nodes directly high above others help in denser environments.

(c) The vertical approach could fail if the operational altitude is restricted.

(d) Utilizing a horizontal setup could enable otherwise infeasible solutions.

Figure 16: Open environments pose no special problems as LOS is essentially guaranteed. In the presence of RF obstacles, using a vertical plane to create multi-hop links between the Control Station and the primary unmanned aircraft provides for conceptually identical solutions. If the scenario does not allow for such a positioning, using a horizontal plane can expand the solution space and enable previously infeasible setups.

As already mentioned in Section 3.2, MVS utilizes a two-dimensional map of the environment, which is suitable to represent most urban settings, as most man-made structures tend to be"2.5-dimensional", i.e. horizontal floor plans extruded "upward," see Figure 9. In the context of communicating in such structured urban environments, the choice of using a horizontal or vertical plane has to be made. From the communications perspective, the main problem to be solved is to allow operations "behind" a building. As an SVS-like point-to-point connection is assumed not to be feasible, a relaying network needs to be utilized. In the simplest case, only one relay is needed, resulting in a total of three Nodes in the network. The positions of these three nodes can hence been seen as the defining points for a two-dimensional plane in the three-dimensional space. Assuming that the positions of the Control Station Node and the Node deployed "behind" a building are fairly fixed, the position of the relaying Node effectively determines the orientation of the two-dimensional mapping plane in space. In the two extremes, vertical and horizontal, the relaying Node is either high and above or low and to the side of the other Nodes. By principle, the vertical high and above option would require the relay to be higher than any obstacle in between the Control Station and the Node that is trying to go behind an obstacle. However, as the

they hardly ever can be used as even with the 20 MHz wide channels there are only three non-overlapping ones.

position of the Node behind the building is mainly driven by operational considerations, it most likely will have to be close to the building. This requires the relaying Node to also move to the far side of the obstacle as to maintain LOS. Figure 16(a) depicts this setup.

If the Control Station is also required to be close to a building, the relying node will not be able to maintain LOS to both lower nodes; a second relay is required. If there is no restriction on the operational altitude (and the range of the utilized links is sufficient), this setup consisting of four nodes can be used to establish a link between any two points lower to the ground (i.e. the Control Station and the Node operating "behind") by simply placing two relay nodes directly above the lower nodes (Figure 16(b)).

Problems arise if the reachable or allowable altitude is not sufficient to allow the high-flying relay Nodes to establish an LOS link. The vertical method fails in this case as the allowable spaces of the Control Station and the Node deployed "behind" are not connected: the hatched area that contains the node "behind" the building is bounded by the ground at the bottom, by the building to the right, and the altitude ceiling at the top, Figure 16(c). There is no connection to the area containing the Control Station and the relay. Switching to a different plane is hence unavoidable and the horizontal plane is most likely guaranteed to be connected. Figure 16(d) depicts one choice of a communications network established in such a horizontal plane.

It is noteworthy that the principle of establishing a communication link between the Control Station and the other deployed Nodes is always the same, regardless of plane orientation: finding LOS paths through a two-dimensional environment. As such, the underlying orientation of said plane does not matter at all and albeit three-dimensional data for buildings can be assumed to be commercially available (compare Figure 8), satellite imagery is even more readily available,[16] making a horizontal plane the easier choice, as

[16]Satellite imagery of a decent recency is often available even for free, although those images are not necessarily ortho-rectified. If COTS data of an area is not available, a system could possibly generate it shortly before a mission through a (higher altitude) overfly of the estimated operation area. However, as the aerial imagery captured from these lower altitudes would contain a certain amount of parallax and perspective distortion, it might be necessary to ortho-rectify them. But as the process of creating orthophotos requires

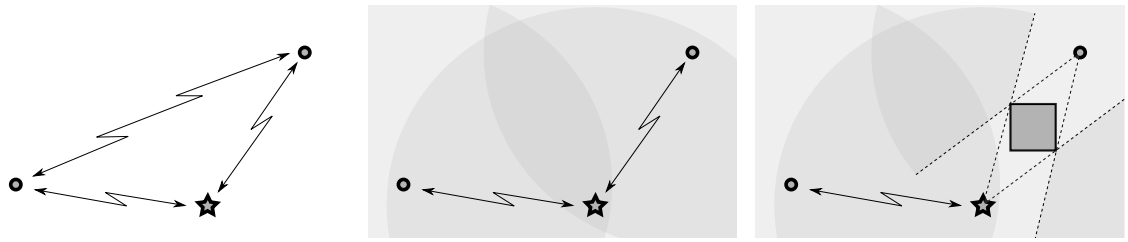
the free/occupied map can with that choice be simply generated from a single and simple data source,the–potentially ortho-rectified–satellite image, whereas the creation of a vertical plane would require the computation of the intersection of the two-dimensional plane with the database holding the COTS three-dimensional obstacle data. Using horizontal two-dimensional maps also allows capturing the interior of most artificial structures, which often also are two-dimensional floor plans extruded "up". Comparable to the prolific satellite imagery, interior maps also started to become readily available publicly (Figure 9).

No matter the method used to create the two-dimensional map, the Voronoi-based maximum clearance route graphs can help to mitigate possible inaccuracies resulting from manually outlining obstacles on non-ortho-rectified imagery or inaccurate obstacle data and complement using a "rough map."

3.3.2 Range-limited Line-of-Sight Communication

Based upon the assumption that datalinks utilized by MVS are range-limited and LOS-restricted, a rather simple technique is used to simulate them: if two nodes are less than a given threshold apart and have a clear line of sight between them, then they can communicate; otherwise, they can not. This approach, sometimes called a Δ-disc method, is used fairly often in conceptual simulations for multiple vehicles as a binary decision aid to determine if two nodes can communicate without the need for an underlying simulation of RF propagation; all that is nee MVS uses a binary method, meaning that the link is either established at full bandwidth or not at all. MVS additionally introduces RF shadowing through obstacles. Figure 17 shows the normally circular Δ-disc is not only limited by range, but also by LOS constrains in the (two-dimensional) environment. To discriminate between the two, the term *RF polygon* is used to refer to the LOS-constrained Δ-disc. As such, the RF polygon method is a very simple way to decide whether a connection can be established. It can be seen as a truly simplistic RF propagation simulation which would put

to overlay images and use a process comparable to stereo-from-motion, that overfly might as well directly create a DEM model of the operational area.



(a) No restriction in RF propagation leads to a fully connected network, i.e. every Node can talk to every other Node.

(b) Range limits eliminate connections between Nodes that are not in each others Δ-disc. Relays are positioned inside Δ-disc intersections.

(c) LOS limitations create the RF polygons; Nodes that are not in each others RF polygons cannot communicate.

Figure 17: In lieu of a sophisticated RF simulation, simple geometry-based link/no-link decisions can be implemented based upon the distance of the sender and the receiver and an assumed discrete range limit as well as the existence of a direct line of sight in the presence of RF-opaque obstacles.

it on the OSI model layer one. MVS uses Obermeyer's VisiLibity [71] to implement the LOS-related computations.[17]

As mentioned in Section 3.1, MVS uses a COTS DDS back end to realize its internal communications which works in a multitude of setup and configuration scenarios. While this is beneficial as it just works, this also means that interfering with it to realize even a simple "simulation" of a communications network (i.e. the OSI model layers above layer one) would require considerable efforts to change and maintain third-party code to which source access is not necessarily guaranteed. Additionally, as DDS operates on the OSI model host layers seven (application) through four (transport), the layers two and three would still be missing. Many MANET protocols make use of location information to improve routing, e.g. [59]. Routing happens in layer three and therefore much interesting UAS communications related research still could not be done. In order to do so, an integration with an actual network simulator, like `ns-3`, seems unavoidable to do research on MANET/GNC interactions.

[17] VisiLibity primarily establishes a visibility polygon, i.e. the area visible from a certain point in an environment, but without a range limitation. From a computational perspective this leaves two options to compute the RF polygon: first compute the Δ-disc in the global environment and then compute the visibility polygon within this sub-environment or first compute the visibility polygon and then the Δ-disc. No actual performance comparison of the two has been done and MVS implements the former.

Given those limitations, as well as the requirement that for SITL and HITL setups the simulation engine would need to be able to communicate with all participating entities anyways, MVS implements the RF polygon-based "network simulation" as part of the actual application, in the OSI model layer seven. What MVS does is essentially fairly simple. When a Node receives a message from the DDS interface (all Nodes should receive all messages as they are currently broadcast), the Node checks in a private method whether this message should indeed be receivable by checking its connectedness to the sending Node. If it is deemed receivable, i.e. when there either is a direct LOS connection or a sequence of LOS hops over relaying Nodes between the sender and the recipient, the message is given to the (protected) processing routines of the Node and its derived classes.[18] If an actual network simulation were to be used, the RF polygon related activities performed in those private methods would not be necessary as the method would directly interact with the network simulator API.

As this RF polygon-based "network simulation" essentially only replicates OSI model layer one, it implicitly assumes that routing will automatically work, that packages make it from the sender to the recipient within an acceptable time, and that they are not dropped or affected by anything else that happens on the OSI model layers two through six. In the presented implementation for MVS, this holds as the utilized COTS DDS solution utilizes user datagram protocol (UDP) datagrams in an internet protocol network to effectively broadcast all messages to all participating Nodes.[19]

3.3.3 Communication Graph

In MVS, the above-discussed connectivity state between Nodes is represented in code and in the GUI through the com graph, which indicates the ability of two Nodes to communicate

[18] This behavior can be affected through a CMake flag, `WITH_RF_SIMULATION`. The flag's behavior, however, is most likely unintuitive so do check the source documentation for more details.

[19] This holds independently of whether a message is a *broadcast* message or an *addressed* message as those distinctions are only relevant within MVS. For DDS purposes all Nodes have DDS Participants connected to each DDS Topic, creating a situation in which all Nodes principally receive all messages.

with each other as an edge between the vertices representing the sending and receiving Nodes. The com graph can be seen as a representation of the (physical) network topology. Checking whether a message is receivable can be reduced to simply checking whether there exists a route from the sender to the receiver in the com graph. Connectivity of two vertices can very quickly be determined through a Dijkstra search,[37] though maintaining currency of the graph requires a little more effort.

Message-triggered interactions with the graph can as such be classified as either checking for receivability of the message such as checking the connectedness of two vertices, or as updating the graph when changing vertices and edges. As the graph's topology in the current implementation of MVS is primarily dependent on the location of the Nodes in the environment, the only messages that are useful for updating are those that convey position information. Upon receipt of a position information containing `State` message, the process outlined below is followed to update the com graph. In this example "host" refers to the node processing the algorithm, "sender" is the transmitting origin of the message, "recipient" is the intended receiver of that message. The host can be the recipient but does not have to be. The process also does not change whether the message is an addressed message to the recipient or broadcast and all nodes are recipients.

Figure 18 depicts this updated process, although the actual behavior of the code implementing it is slightly different as some corner cases are considered.

1. If the sender has moved further than a given threshold,[20] disconnect the sender and thus delete all edges connected to the com graph vertex representing the sending Node in the host's com graph.

2. Determine all connected components of the com graph. There have to be at least two–disconnected sender and the rest–but there could be more either because the

[20]For the purpose of reducing computational complexity, hosts maintain two positions for each Node in the network. One is always kept up to date and is associated with the Node itself, the other is associated with the vertex representing that Node in the com graph. If the distance between those two breaches a threshold, the com graph update is triggered.

graph has been disconnected before or because the sender was a separating vertex between clusters of vertices.

3. Determine the RF polygon for the new position of the sender.

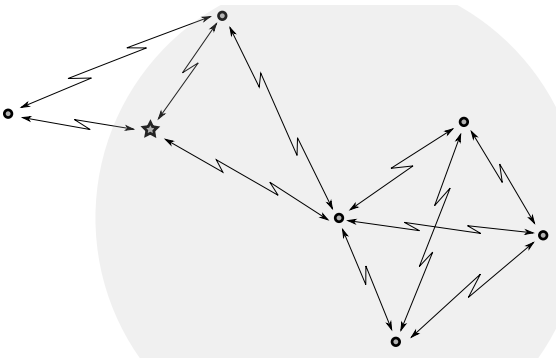
4. Check whether other nodes (i.e. all vertices in the connected sets of interest but the sender) have a position that lies within the newly computed RF polygon of the sender. If a node is within the RF polygon of the sender, add a corresponding edge to the com graph.

Of particular interest is the question how to treat messages that are send by Nodes more than one hop away. Issues in this context immediately touch upon the simulation of the underlying (physical RF) network infrastructure and how that relates to the routing of packages. One the one hand, Nodes should not need to care about how messages arrive, and having a better knowledge of the network topology could be beneficial for overall mission performance.

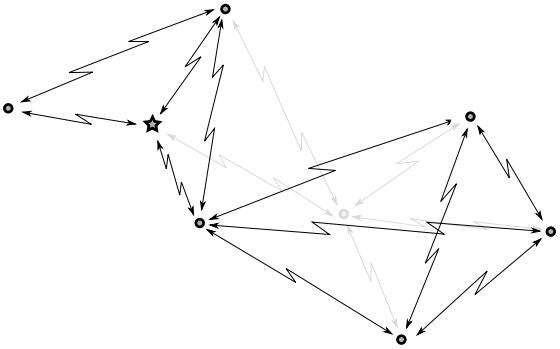
Closely coupling these two–network routing and GNC algorithms–might be beneficial, if GNC algorithms would want to make use of the network topology–which at this point would provide useful information "for free" such as which node is the nearest neighbor. The presented implementation of MVS could not look into this with sufficient detail as no good enough simulation of the underlying network was implemented. This would have been necessary, however, if one considers the following related aspects:

- If the network is not simulated[21] and the Nodes have no access to packet routing information, then no meaningful com graph can be build. The only graph possible would be a star with the hosting Node in the center. All received packages, either from a one-hop neighbor or forwarded through the network, would imply a direct

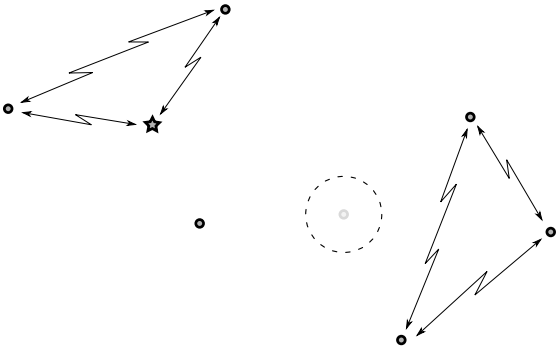
[21] Maybe it would be better to use the term *estimation* here as opposed to simulation. What is meant is an on-line process running on the host that creates the com graph–independently of whether the used network is real (e.g. during a flight test or an actual mission) or simulated through, for example, `ns-3`.



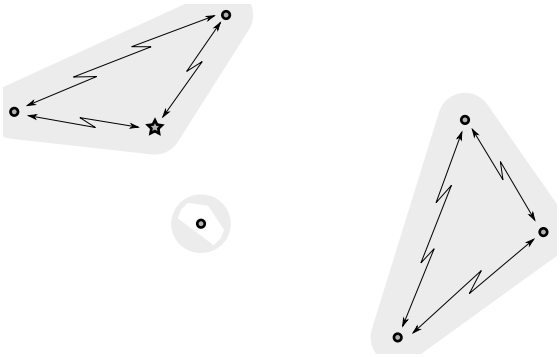
(a) A com graph of a network showing the Δ-disc of one Node. The graph is connected, i.e. each Node can communicate with all other Nodes, possibly using relaying Nodes.



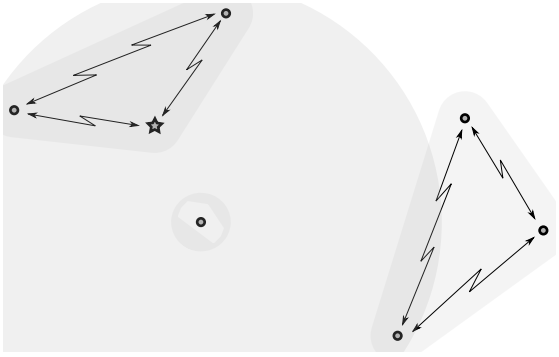
(b) Based upon a received position update, the location of the sender's vertex is updated in the host's environment. Edges, defined between vertices, are not affected by vertex position changes.



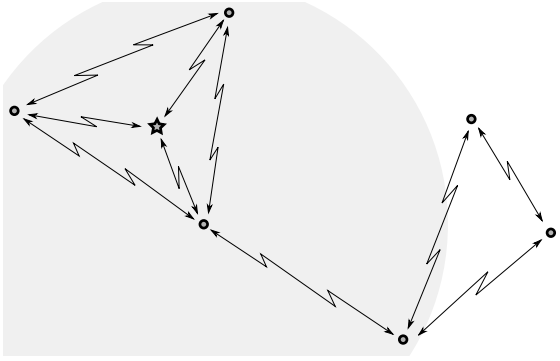
(c) As the position change exceeds a threshold value, affected edges are invalidated and deleted; the sender is disconnected from the com graph.



(d) The resulting connected sets are identified as in certain cases only the host's set is of interest. The sender always will be in its own set.



(e) The Δ-disc for the new position of the sender is computed. (In the presence of obstacles the RF polygon would be computed.)



(f) All Nodes that are part of a connected set of interest (all in this case) and that are inside the Δ-disc are reconnected to the sender.

Figure 18: The com graph update process upon receipt of a message containing position information of its sender.

connection between the sender and the receiving host as routing information is lacking. Depending on the actual routing mechanism used, this situation would also allow cases where the created network map might differ amongst Nodes participating in the same network as they might or might not be exposed to different messages.

- If the network is not simulated and the Nodes have access to packet routing information, then the Nodes could create a com graph-based upon the evaluation of the routing data in the received (or intercepted) messages. This would only allow a correct mapping of the "upstream" routes of the com graph which actually propagate through the host. As those routes would be guided by the unknown routing mechanisms, this mechanism also could lead to the situation in which different Nodes could have different topology maps of the same network.

- If the network is not simulated, but the Nodes have access to routing information and the utilized routing scheme builds or requires a global map of the network at least sometimes, then all Nodes could build a correct topology map of the network.[22]

- If the network is simulated, correspondence between the assumed/simulated com graph and the actual underlying network topology (the physically possible links) is heavily dependent on the utilized routing methods and the hop range of broadcast position messages as host Nodes obviously would need up to date location information to predict if a connection between a sender and a receiver is actually possible. The extra computational effort to do this, however, could increase the overall system's performance as individual Nodes could potentially predict network coverage and as such tolerate operations which do not require constant connectedness to a human operator.

[22]This situation would seem to be beneficial as no additional computational effort would need to be expended by the host to build the com graph. However, this clearly shows how a choice in the GNC context of UAS operations can affect the set of MANET protocol candidates. Several location aware mechanism have been proposed, though not all of them build a global map at every node.

The problem of correctly mapping, estimating, predicting, or simulating the network topology online on a Nodes seems to be comparable to an observability study. The tightly coupled integration of GNC algorithms with the network routing schemes could then possibly be phrased in a controlability framework.

Related is the question how to build the com graph in SITL and HITL setups as in those cases Nodes can receive messages which they should not be receiving according to the simulated scenario. The presented implementation of MVS is such a case as all participating Nodes receive all messages (broadcast or addressed). For research this seems to be a beneficial setup as a Node, for example, the Control Station of the researcher can present all information available in the simulation framework.

Another issue is that disconnected sets technically are not known to the host processes building the com graph, as "disconnected" means that information cannot be passed between them. However, those sets that had been connected to the host Node prior to disconnecting the sender are known to host's processes. To distinguish both kinds the former type (the previously disconnected sets) will be referred to as *far* sets and the latter kind (the previously connected set) as *near*. All sets in Figure 18(d) are *near* disconnected sets. Problems now arise with these near and far sets in the context of the very simplistic, RF-polygon-based "network simulation" in MVS and its implementation in code as this needs to happen at least in real time. The present code however does not strictly differentiate between the network simulation part (i.e. the private method in a `Node` class that decides whether a message is receivable or not) and the part that builds the com graph for the host; after all, they would both build the exact same graph. Reusing this information allows for a reduced computational cost as the identical steps for "simulation" and "graph building" only have to be performed once.

Not sharing this information could lead to problematic race conditions in cases where a connection to another connected set is made as in such a case a high computational cost needs to be expended to explore the newly connected set and to discover or rediscover the

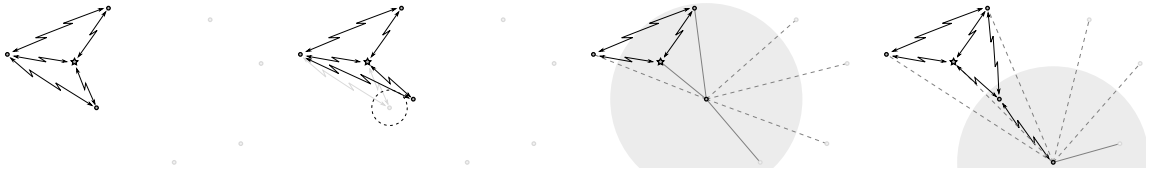
edges internal in that set. As this computational cost can be hard to predict, giving an upper bound on the completion time is complicated. The race condition can now occur if during such an update of the com graph the position of another Node changes.

Comparing Figure 19 and Figure 20 highlights the possible differences in the computational load for the cases where the intra-set connections of a disconnected set are not maintained or maintained and shared respectively. Although no in-depth study has been performed to quantify the actual global load difference between those depicted methods, the setup in which connections in disconnected set are maintained was chosen for implementation in MVS. One apparent benefit is the existence of an upper bound for the complexity of each update step. The computation of one RF polygon and, for N Nodes in the Network, $N - 1$ checks whether a given Node/vertex location is covered by that polygon.[23]

Another benefit of maintaining the connections of disconnected sets is that the researcher can see information related to the overall simulated system, such as a cluster of nodes which do not have a communication connection to the Control Station. Visualizing this is obviously a design choice, and a future version could possibly introduce a dedicated experimenter station which would show the overall situation and a Control Station and Vehicle GUI that only shows what the Node could actually know. This also relates to the question of whether a map of the environment should show movement of Nodes that do not have a communications connection to the host presenting the map. The choice to show this movement and the related "simulated" com graph connections could pragmatically be argued as replicating a very good prediction algorithm of where Nodes are expected to be, given that the last flight plan and the utilized routing algorithms are know.[24]

[23]The minimal displacement threshold guarding a com graph update is indeed introduced to slow down the need of com graph updates as the currently implemented code cannot maintain real time updates in the presence of various Nodes that broadcast a position update at 10 Hz. Without the guard, the number of required RF polygoncomputations scales quadratically as every node needs to compute the RF polygon for itself and all other nodes at every update.

[24]This argument obviously is somewhat flawed in situations that allow for multiple Control Station as in that case another, potentially unmonitored, source of flight plans exist and hence the information feed into the system is not observable any longer.

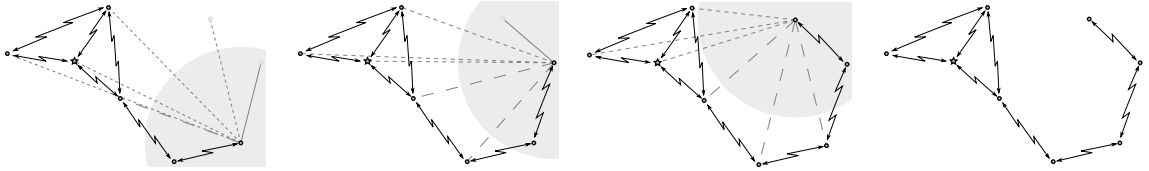


(a) The starting topology with the undiscovered Nodes grayed out.

(b) A (connected) Node crosses the threshold towards the undiscovered Nodes.

(c) Computing 1 RF polygon and 7 "withins;" 3 connections discovered.

(d) Computing 1 RF polygon and 6 "withins;" 1 connection discovered.



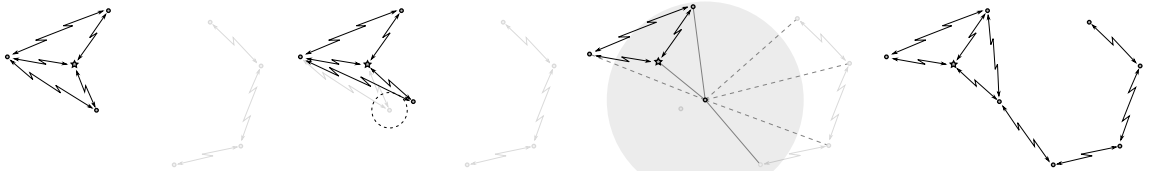
(e) Computing 1 RF polygon and 6 "withins;" 1 "within" redundant, 1 connection discovered.

(f) Computing 1 RF polygon and 6 "withins;" 2 "withins" redundant, 1 connection discovered.

(g) Computing 1 RF polygon and 6 "withins;" 3 "withins" redundant, no connection discovered.

(h) The final topology after the discovery process has concluded.

Figure 19: The discovery process of unmaintained connected sets requires a depth- or breadth-first exploration of the com graph when new connections are discovered. In the shown scenario, a total of five RF polygons and 31 "withins" are computed (six of which are redundant).



(a) The starting topology with the undiscovered Nodes and connections grayed out.

(b) A (connected) Node crosses the threshold towards the undiscovered Nodes.

(c) Computing 1 RF polygon and $N - 1 = 7$ "withins;" 3 connections discovered.

(d) The final topology after the discovery process has concluded.

Figure 20: The discovery process of maintained connected sets requires a fixed computational effort. For a graph with N vertices, one RF polygon and $N - 1$ "withins" need to be computed.

As checking for receivability of a message in the com graph happens more often than updating it and the topology of the graph cannot change between updates, the previously mentioned use of a Dijkstra search to check for connectedness does not need to performed every time receivability of a message is checked. A simple exchange of computation for memory can be made, and the connectedness status of all Nodes in the network can be recorded based upon a single search. When a message arrives that does not contain position information (any message other than the `State` message) no computation needs to be expended to run a however-fast graph algorithm. A simple memory look-up of the connectedness flag suffices.

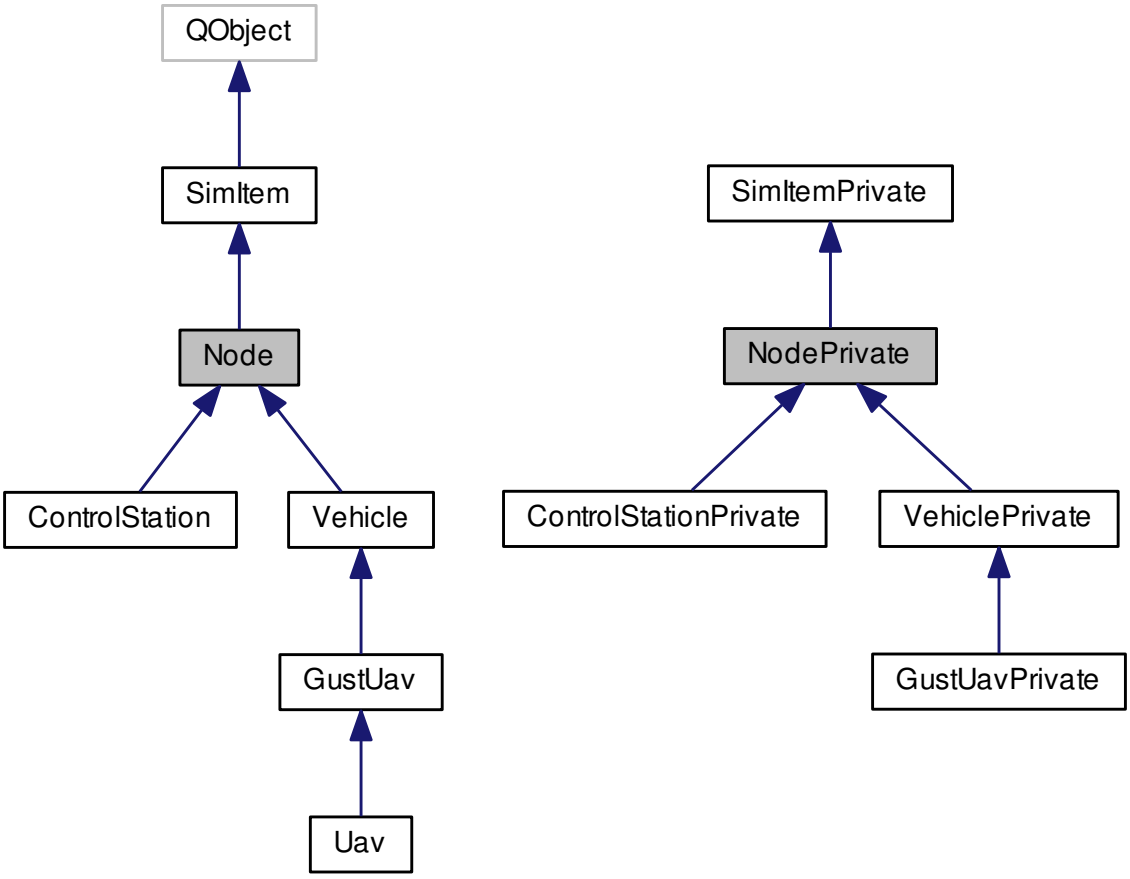
3.4 Code Structure and Programing Interface

The presented framework is primarily coded following an object-oriented approach, using the C++ programming language in the C++11 version. The target development environment was a 64 bit Linux system, using the GCC 4.9 Release Series compilers.[1] The framework also makes use of some well-known, freely available open source packages: Eigen, the Boost C++ libraries (Boost.Graph, Boost.Geometry, Boost.Uuid), and the Qt environment.[36, 49, 81, 88] As already mentioned, the DDS backend is provided through a commercial solution from RTI [74] which was made available through a free academic license. The framework also provides added functionality through the use of Obermeyer's open source VisiLibity [71] and Held's VRONI [52]. The sources to the VRONI package were also made available through a free academic license.

Fore more information about the sources of the used software and their respective licenses see Appendix A.

3.4.1 Object Structure and Inheritance

MVS uses an object oriented approach which is centered around the requirement to enable several executables, possibly distributed among various hosts, to communicate and as such act as a team or network of vehicles. At the core of this idea is the MVS Node class



(a) Inheritance diagram of the Node class.

(b) Inheritance diagram of the NodePrivate class.

Figure 21: The Node class is the core object in MVS. In parallel to the main classes, there is a layer of associated Private classes which provide the necessary private implementation of methods to allow a binary-compatible compilation of libraries.

as the main communication facilitator and DDS interface. Figure 21(a) depicts the main inheritance scheme for allNode types used in an initial human subject study (Chapter 5) on the effectiveness of a placement aid.

Realizing that larger research software projects always have users more focused on software enhancement and feature development while others will focus more on simply using already-established methods, MVS tries to restrict publicly-accessible methods of classes to calls only relevant for a non-developing end-user and correspondingly tries to hide all related and underlying helper and support functionality in private methods so public class methods form a sort of end-user API for MVS. As a result, a frequent change in

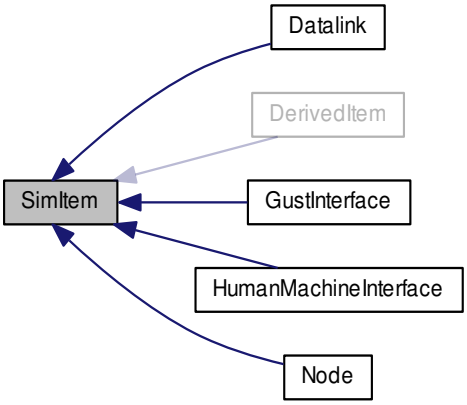
the structure and/or functionality of said private methods is expected and anticipated and the presented framework addresses this through the use of the pointer to implementation (PIMPL) idiom.[25] On the one hand, this idiom effectively hides all private implementation details in a separate class whose name is created by appending Private to the original class's name, accessible through a private pointer which is indeed the sole private member of the public class. In the realm of Qt this pointer is referred to as the *d-pointer*, see [84, 87]. Conversely, the idiom also enables binary compatibility which, although currently not used in the presented code, is of value as it can considerably lighten the burden of version compatibility of shared libraries. For the end-user, binary compatibility means that end-user code, which (dynamically) links to core libraries of MVS, should continue to work without the need of a recompilation in cases where the core libraries are updated; the older library binaries can simply be replaced by the newer ones.

When looking at Figure 21, the use and benefit of using the PIMPL idiom is easily explained. Comparing the top of the two figures, the absence of a QObjectPrivate is apparent. This is not due to the fact that there is no such class (indeed, Qt makes heavy use of the PIMPL idiom), but merely a result of the fact that MVS itself is an end-user of the Qt framework. As such, MVS does not need to care about or access the related Private classes, as MVS only uses the public API of Qt. The same is true when looking at the bottom of Figure 21: the class Uav indeed does not have a corresponding UavPrivate as it is included in the MVS framework as an example for non-development end-user code.[26] As Figure 21 shows, the use of the PIMPL idiom leades to parallel structures throughout the complete inheritance chain of a SimItem.

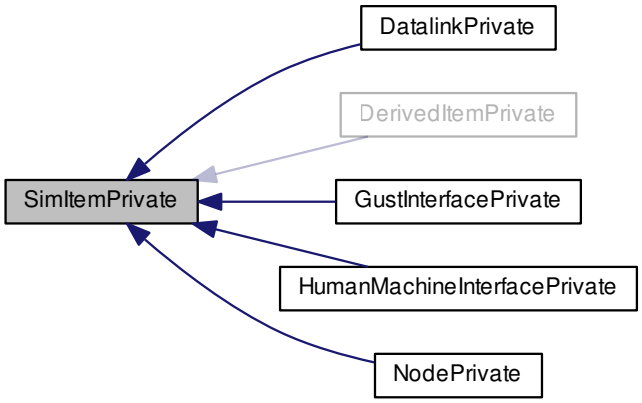
The SimItem class, a parent of Node, is a base class for all items of the simulation that could be considered to be their own "module" of sorts (Figure 22). It also follows the

[25] The idiom is also called "Bridge Pattern" in [43].

[26] In the case of Uav, the code showcases the use of the MathWorks MATLAB Engine to plot the trajectory of a Vehicle via MATLAB. This demo behavior is controlled through the related CMake flag `WITH_MATLABENGINE`.



(a) Inheritance diagram of the SimItem class. (Abbreviated after Node.)



(b) Inheritance diagram of the SimItemPrivate class. (Abbreviated after NodePrivate.)

Figure 22: The SimItem class provides the fundamental instantiation interface for various modules of MVS. (DerivedItem and DerivedItemPrivate are grayed out as they are only examples used for in-code documentation.)

PIMPL idiom and provides the basic simulation controls *initialize*,*run*, and *halt*.

Unfortunately the currently provided interface for the initialization, running, and halting is not as streamlined as it potentially could be which is mainly related to the fact that SimItems are used in inheritance ("is a"-relationship) as well as composition ("has a"-relationship) of classes, potentially accross the public/private boundary introduced through the PIMPL idiom. For example, Node is a SimItem, but NodePrivate also has a Datalink, which itself is a SimItem. In the same manner, ControlStationPrivate has a HumanMachineInterface and GustUavPrivate has a GustInterface. It would be convenient to have the SimItem::initialize() method to propagate down through the inheritance and composition levels through the use of virtual function lookups and, once reemerging back at the SimItem level, set the SimItem::isInitialized flag and be ready to wait for the SimItem::run() method call, but as the SimItem class also provides the glue for all the state machine-related interfaces (Section 3.5.2), this does not work. The reason being that SimItems attached via composition do not spawn their own state machine but nest themselves into their parent's, which would need to have been initialized to allow this as otherwise no root state would be present to nest into. As such, the initialization routine does require some manual sequencing of the calls to parents' initializeItem() method.

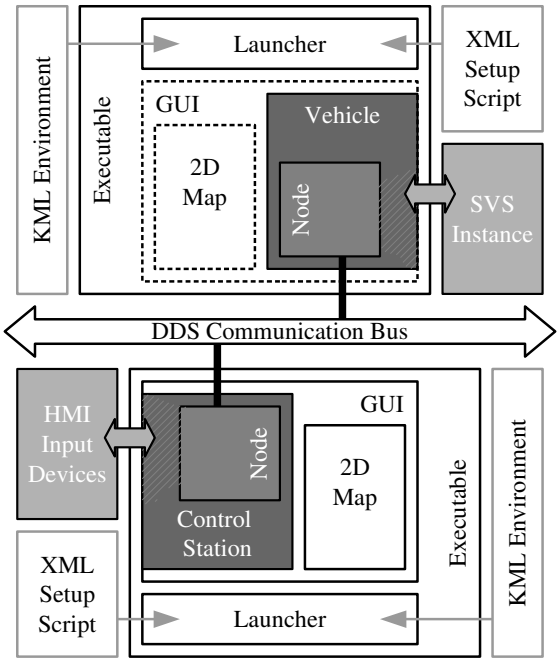


Figure 23: Initialization and setup of an MVS executable is performed by a dedicated launcher part. The top part represents a console executable associated with a vehicle; the bottom part represents a GUI application associated with a MVS control station and the affiliated human-machine interface input devices.

3.4.2 SVS Instance Interfacing

When comparing Figure 21(a) and Figure 23, the actual mechanisms to create a Vehicle become a little more apparent: Vehicle provides all the necessary/required public methods that an end-user could or would need in the form of purely abstract methods. Uav, for example, uses these methods to get the data necessary to plot the vehicle's path in MATLAB. GustUav is then merely a class that provides the actual implementation of Vehicle's abstract methods as those are specific to the actual SVS instance in use and provides the direct interface to the utilized SVS instance. Correspondingly, GustUav is represented in Figure 23 as the shaded section of the Vehicle block that deals with the SVS communication (represented through the double headed arrow connecting the SVS instance and the vehicle block).

In the case of the presented MVS implementation, all vehicles are simulated through

the Georgia Tech UAV Simulation Tool (GUST), an advanced software- and hardware-in-the-loop GNC research tool which can handle the complete spectrum from SITL simulation to flight testing with various aircraft and airframe configurations. As previously mentioned in Section 3.1, MVS only requires a minimal interface with an SVS instance. In the case of GustUav, MVS only interprets three down- and three uplink messages.[27]

3.4.3 Setup, Initialization, and Launch

In an effort to make MVS usable in various usage scenarios, possibly even running directly onboard flying vehicles, the framework is made up of individual executables, each representing one or more Nodes in the Network (Figure 6). The framework's source code can either be compiled into a smaller non-interactive application for console-based execution or a larger interactive application that provides a GUI; the choice between them is a setup option chosen by an end-user of MVS.

Further customization and setup options are governed by setup scripts, tailored to each individual executable. These scripts, written in extensible markup language (XML), are passed as an argument and parsed upon startup by a dedicated launcher part within the MVS executable, be it the console or the GUI version. This launcher initially validates the passed-in script against a XML schema definition (XSD) and then proceeds with loading the environment, setting up the Node(s) affiliated with its executable, and configuring the communications between the Node(s) and the associated SVS instance(s). Figure 23 gives a high-level overview of the parts involved in creating MVS executables and Nodes.

The setup script contains the following information:

- The position of the map datum used by the executable,
- the location and shape of a geo-fence,

[27] `datalinkMessage0`, `datalinkMessage1`, and `datalinkMessageFlightPlan` for the downlink from the GUST vehicle instance and `datalinkMessageUp0`, `datalinkMessageFlightPlan`, and `datalinkMessageTrajectory` for the uplink to the GUST vehicle.

- the file containing obstacle data for the environment,
- and the setup data pertaining to eachNode represented by this executable.

The launcher uses that information to build an environment from the given geo-fence and the obstacles, instantiates a Node, feeds it that environment, and prepares it for a connection with its affiliated SVS instance. To do the latter, the Node specific section(s) of the setup script contain the following information:

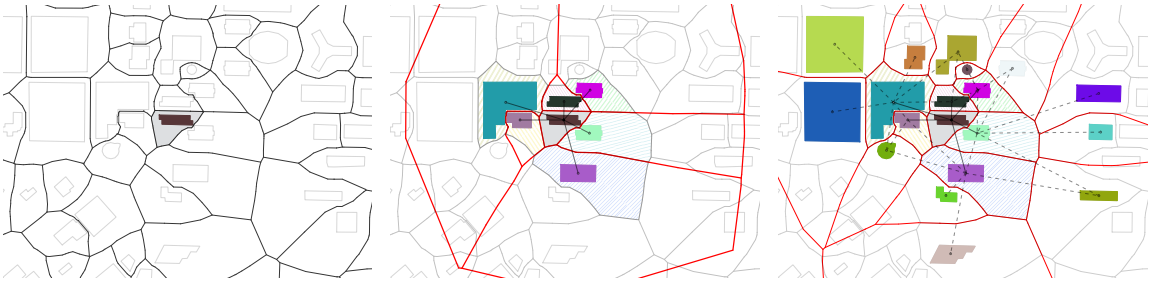
- The type of the node (i.e. Vehicle or Control Station),
- a (not necessarily unique) name of the Node,[28]
- the location where the Node is to be initialized,
- and a delay determining when the Node is to be initialized.[29]

Depending on the Node type, various other specialized data are given. Control Stations, for example, specify human-machine interface (HMI) data, such as joystick calibration data and button assignments, whereas Vehicles specify the details necessary to establish the communication to their respective SVS instances and can provide another mapping for joystick input data in order to homogenize how the vehicles "feel" to a human joystick operator.[30] The latter requires that Vehicles be further specified so that both the XSD/XML sections of the setup scripts, as well as the code-internal interface and translation parts, can be adapted to the requirements of the specific SVS type. (See Section 3.4 for more details on the underlying architecture of the source code.)

[28] MVS automatically generates globally unique identifiers to identify Nodes internally.

[29] The startup delay, expressed in ms, is also used to create a primitive load balancing as it allows to desynchronize the 1 Hz, 5 Hz and 10 Hz loops of Nodes being executed on the same host.

[30] As MVS is intended to work with heterogeneous teams of vehicles, operators could face various challenges when switching. One of which can be changes in vehicle responses and responsiveness to manual joystick inputs. Cases to be considered could be switching from rotary to fixed wing aircraft categories as well as switching from smaller and dynamically faster to larger and slower types of the same category.



(a) A single obstacle cell and the complete route graph for a larger environment.

(b) The 1-hop neighborhood and the route graph computed only from those obstacles (and the boundary).

(c) The 2-hop neighborhood and the route graph computed only from those obstacles (and the boundary).

Figure 24: Computing the Voronoi partitioning for a reduced environemnt provides results that are very close to those for the full environment.

3.4.4 Dynamic Environments

Being based on a Voronoi partitioning, the individual segments of the route graph (the straight line segments connecting the waypoints along a route between bifurcation points), are only affected by the obstacles closest to them. The reduced environment of the current obstacle cell and its direct neighbors (Section 3.2.2.2) is sufficient to determine the perimeter of the current obstacle cell as by design only the closest elements affect the Voronoi diagram. To a certain extend, this reduced environment also describes the routes forking from the obstacle perimeter sufficiently well, as at least close to bifurcation point from the perimeter of the current obstacle cell, only its one-hop neighbors affect it.

Figure 24(b) shows a route graph computed for the one-hop distance environment of an obstacle in red. The graph matches the route graph computed for the complete environment (shown in gray, compare Figure 24(a)) for the perimeter of the cell. This result is to be expected as all obstacles influencing those route graph segments are part of the reduced environment. As postulated above, the route graph for the reduced environment also matches the routes branching from the bifurcation points of the cell extremely well; only a very close inspection reveals minute differences on the routes leaving towards the North and the South-West. If the two-hop-distant obstacles are included in the reduced environment, those paths are also postulated to match. In the case of the two-hop-distance

environment, all boundaries of the one-hop-distant cells are guaranteed to match the routes of the complete environment as all route affecting obstacle polygons are included.

The main benefit of this relation is the intrinsic limitation how "far" through the Voronoi diagram the altering effects of a change in an obstacle propagate. This can be helpful in a future expansion where Control Station operators only need to roughly outline an obstacle, relying on SLAM capable Vehicles to update the map with the exact shape of encountered obstacles. The resulting changes in the traversable route graph are localized, which should allow a decent scalability.

3.5 State Machines

One research interest related to MVS can be called the "single operator aspect:" how can a single human operator maintain meaningful control over a team of unmanned aircraft in a way that increases performance despite the added management and control overhead caused by the team? Simply, this involves making sure the operator can always answer the three famous questions *What is it doing?*, *Why is it doing it?*, and *How do I change it?*

The three questions posed above all relate to the behavior of *it*. Colloquially this *it* is the unmanned aircraft, its automation, and the autopilot. More detailed and in the context of MVS, *it* can be argued to be mainly reflected by the guidance portion of the GNC suite determining the "behavior" of the unmanned aircraft utilized in MVS. In a potentially heterogeneous MVS though, the GNC suites utilized by the various unmanned aircraft from potentially different SVS are not necessarily the same and are by design hard to observe for a human operator who is not necessarily trained in the intricacies of all the systems. Nor might the operator actually have access to the data affecting the GNC algorithms as MVS, in order to allow the use of a variety of SVS aircraft, blocks direct access to that data through a layer of abstraction, as outlined in Section 3.1.

Given those limitations, an apparently necessary condition to achieve observability and controllability of the unmanned aircrafts' "behavior" would be that all behavior-affecting

data is available through MVS, and by extension to the operator, and that behavior-affecting decisions are made within MVS. This consequently requires that the inter-system API from MVS to the utilized SVS aircraft can transport all the corresponding information without limiting or cropping the intended meaning. The datalink requirements listed in Section 3.1 describe which information this inter-system API can transport; an allowed and very simple interface is the exchange of position (downlink) and flight plans (uplink).

Flight plans can be seen as an answer to the "What is it doing?" question: the unmanned aircraft is going from some place (first waypoint in the flight plan) to another place (last waypoint in the flight plan), and is doing that in the described way (all the interim waypoints in the flight plan). Obviously there is a level-of-detail issue intrinsic to all of the three questions, as, for example, the "What is it doing?" question for an unmanned aircraft powered by an internal combustion engine could prompt the answer "enriching fuel/air mixture" if posed during landing preparations. A more relevant answer might be "descending." A flight plan certainly is not an all-encompassing answer to the "What is it doing?" question, but if the flight plan is describing a planned trajectory, i.e. the waypoints also provide information when the unmanned aircraft is supposed to be there, then such a plan could provide enough insight into the current waypoint and future state of the unmanned aircraft to be of use.[31]

What remains to be answered are the *Why is it doing it?* and *How do I change it?* questions. These questions can be slightly rephrased into "How did it get to this mode?" and "How do I get it out of this mode?" if *mode* is used to capture a more holistic description of the overall unmanned aircraft system.

These questions, as well as the "What is it doing?" question, can be answered through

[31] How to describe and convey the information contained in a trajectory-describing flight plan to an operator is a completely different question though, especially if the flight plan allows waypoints to contain not only position, but also attitude and possibly even rate information. Interesting questions are related to the feasibility of flight plans describable with such detailed waypoints and how to deal with contingencies resulting from missing all or part of the requirements of a waypoint.

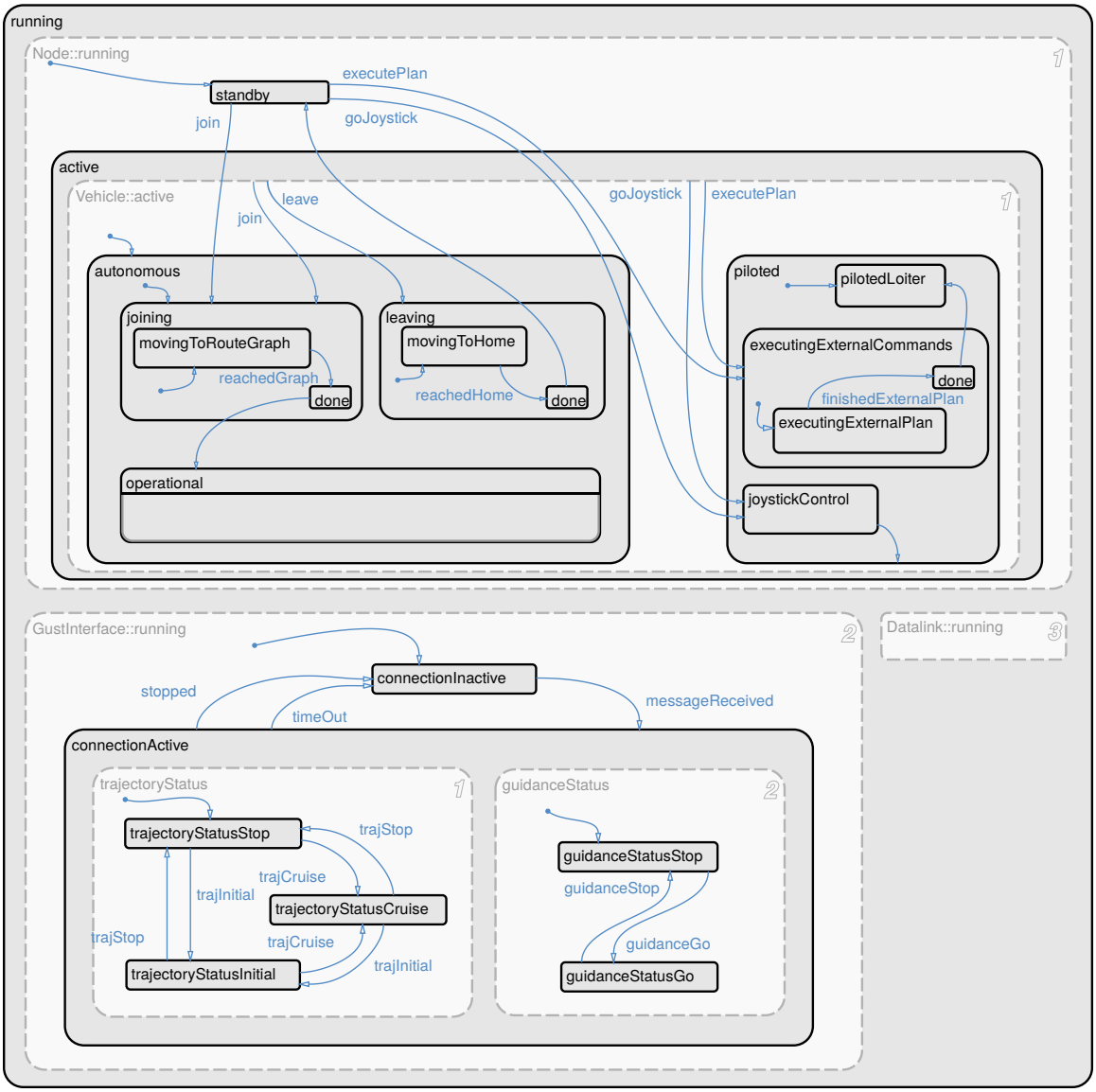


Figure 25: State chart of a GustUav Vehicle. States are represented as labeled boxes; their hierarchy is indicated through nesting. Solid outlines mark the state as being exclusive (the system cannot be in any other state at the same hierarchical level). Dashed outlines mark states as being parallel (the the system is always in all parallel states of the same hierarchical level). Events that cause a transition from one state to another state are indicated through blue arrows, connecting the source and the target state. Event arrows starting at a dot indicate the default transition when the surrounding state is entered.
(For more details on state chart graphics see [50].)

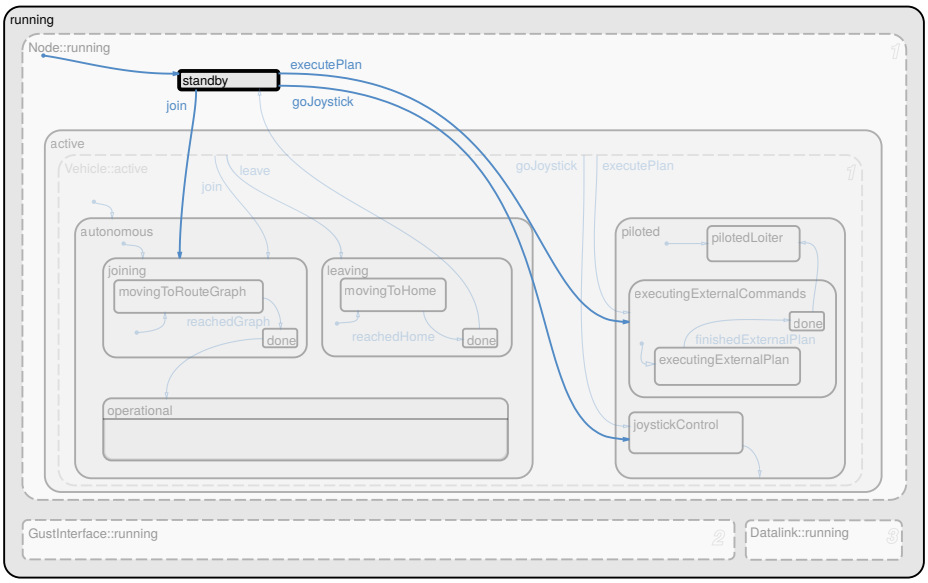
visualizing an underlying state machine which determines the *mode* of an unmanned aircraft. Although MVS does not currently support a visual representation, the potential to do so, as well as the related benefits during writing and debugging, was a major consideration to represent the behavior governing system in MVS in a dedicated event driven state machine. Figure 25 shows a state chart of a GustUav, generated using MathWorks' Stateflow.[32][63] The graphic closely follows Harel's original description of state charts [50]; the main difference is that parallel states are enclosed in dashed boxes so that they could be labeled as well.

3.5.1 Reducing Mode Confusion with State Charts

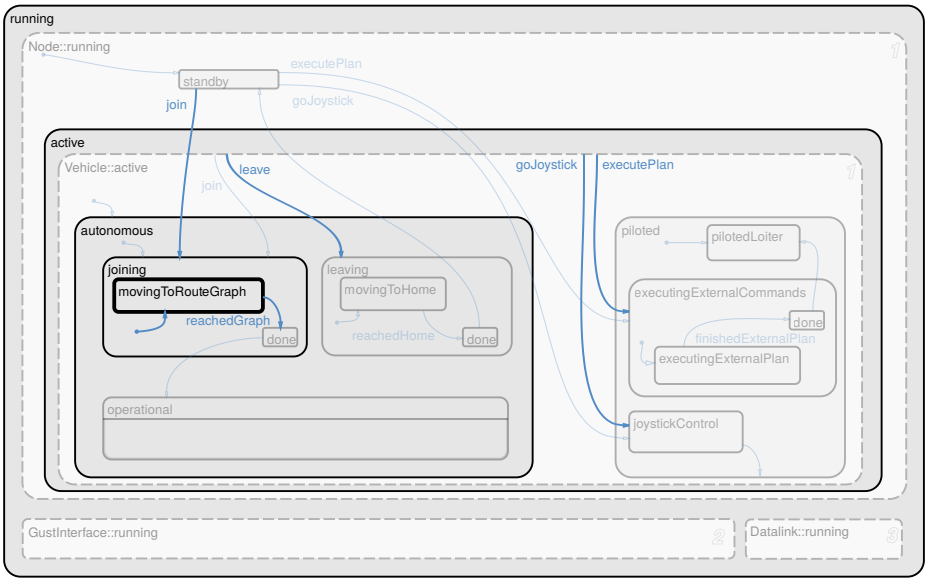
Figure 26 gives an example how the *What*, *Why*, and *How* questions can all be answered through a state chart: what the system is doing can be indicated through highlighting the current state, why it is doing that through marking the transition event that got the system into the current state, and how to change it can be visualized through all possible events that move the system out of its current state.

State charts also provide a means to approach the mentioned level-of-detail issue as they encapsulate detail in the levels of hierarchy. Using Figure 26(b) as a example, the answer to the "What is it doing?" question can be given in various levels of detail. Primarily the vehicle is running; increasing the level-of-detail, the answer can be qualified as the vehicle being active, autonomous, joining, and moving to the route graph. Although this does not get all the way to changing the mixture of the engine, in theory another parallel state in running could describe the state of a full authority digital engine control (FADEC) system governing the engine parameters.

[32] Stateflow was only used to "draw" the state chart during the development process of MVS's state machine. Stateflow is embedded into Mathworks' Simulink and as such does support automatic code generation and an online connection to (simulated) systems, although none of these features are currently integrated with MVS.



(a) The current state standby has been reached through a default transition from the parallel state SimItem::running. Possible exit transitions are join, executePlan, and goJoystick.



(b) The current state movingToRouteGraph has been reached through the join transition from standby. Possible exit transitions are reachedGraph, leave, executePlan, and goJoystick.

Figure 26: A state chart representation of a typical start up process. The Vehicle is booted and then reaches standby. After a join command has been issued, it moves toward the route graph.

3.5.2 State Machine Coding

The level-of-detail issue has another aspect in how the state machine is integrated into MVS's code framework. As every computer can be represented as a state machine, every computer program can also be pressed into a state machine framework, meaning that the complete code making up MVS could potentially be represented in a state chart. The level-of-detail question is which parts of the source code should have a direct analog in the state machine that governs the behavior of Nodes (i.e. the state chart shown in Figure 25) and which parts should not.

On a code level, this distinction can be made a little clearer by differentiating between the "normal" (normally single threaded) procedural flow of MVS code and the "event triggered" (normally multi-threaded) transitions through the state machine. As MVS utilizes the state machine framework provided by Qt, which neatly dovetails with Qt's native thread-crossing signal and slot mechanisms, triggering events can be easily accomplished and the state machine also operates somewhat detached from the main working threads. The learning curve coming from "conventional" procedural coding can be quite daunting. As MVS from the beginning was intended to support both, more framework oriented developers as well as researchers more interested in just using it, a medium needed to be found. Of particular interest in this context are states that can be associated with *activities*: states that are directly related to a task which has a clearly identifiable end. Figure 25 shows three such activities: joining, leaving, and executingExternalCommands. Activity states are descriptively labeled with a verb in progressive form (other states use adjectives as labels) and have at least one "action" (what the activity is doing) and a done state as sub-states. The action for joining is movingToRouteGraph; the one for leaving is movingToHome, and for executingExternalCommands it is executingExternalPlan. These action states have a direct analog in the "normal" procedural code of MVS and are meant to be easily serviceable code (useful for those researchers), whereas the integration of of those actions into the state machine framework is better suited for developers interested in the state machine

framework of MVS.[33]

The "procedural" process of entering the movingToRouteGraph state are fairly straightforward. Upon entry of the state, the closest point on the route graph with respect to the ownship is determined and then used as the target waypoint for a flight plan directly to that point.[34] The utilized function call `goTo_direct` creates a flight plan, uploads it to the attached SVS instance, and triggers its execution. As currently all action states involve the execution of flight plans, a "user" coder does not have to worry about detecting when the just-started flight plan is finished as the state machine framework will keep track of that through GustInterface::running, a parallel state, seen in the bottom of Figure 25. If a flight plan is finished and the Vehicle is still in the action state movingToRouteGraph, the event reachedRouteGraph is triggered, and the Vehicle's state machine transitions through the done state into the operational sate.[35]

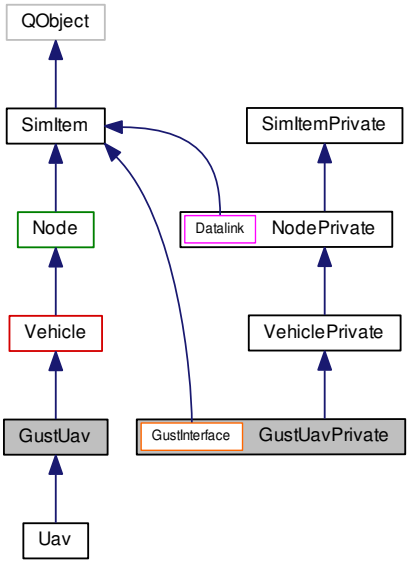
The state machines also have to fit within the object inheritance structure of MVS. As previously mentioned in Section 3.4.1, SimItems are simulated items that can be considered to be their own "module" of sorts. With the introduction of state machines, this can be clarified as SimItems are simulated items which could benefit from providing a state machine. As such, SimItem is set up to provide an instance of a Qt state machine. However, as there seems to be no apparent benefit of operating several stand alone state machines in parallel on one Node, MVS is set up to utilize the parent's state machine in case of inheritance or composition of SimItems.

As depicted in the inheritance diagram in Figure 27(a), a GustUav is a Vehicle which is a Node and also a SimItem. Figure 27(b) color codes states and event transitions with the classes that are responsible for them: the nesting in the top of the state chart reflect the

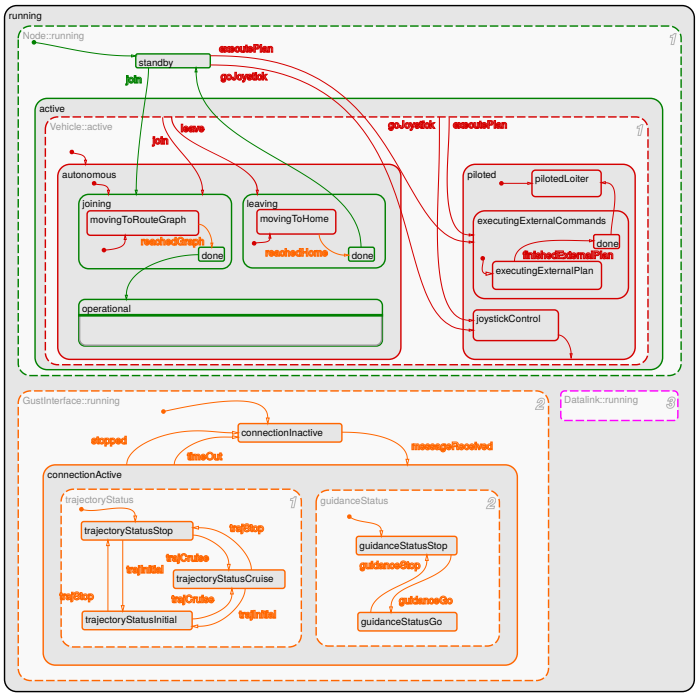
[33] Although considered to be "user serviceable," the related code is however still part of the VehiclePrivate class and as such not accessible to MVS users who are only given pre-compiled libraries and API headers.

[34] The (two-dimensional) environment map only gives the (x, y) parts of the coordinate triplet, the z-coordinate is gathered from the predetermined operational altitude of the vehicle, i.e. the altitude for which the environment map is valid.

[35] The done states are a somewhat technical (Qt) necessity. Entering them will automatically trigger the event leaving the state, which is why none of the event transition arrows leaving the done states has a label.



(a) A combined inheritance diagram of GustUav and GustUavPrivate, also showing the composition with other SimItems.



(b) The GustUav state chart color coded by the creating class of states and transition events. Class inheritance is visible at the top in nested states, the bottom shows composition effects through parallel states.

Figure 27: Comparing the inheritance and composition of the GustUav object and the related structure of the state machine running in GustUav.

inheritance, and the parallel states in the bottom the composition. In this example, the GustUav's SimItem parent provides an instance of a Qt state machine as it is the first SimItem to be constructed. This SimItem, providing the state machine instance, also provides the root state running, the outermost black box in the state chart. To prepare the class for composition, the SimItem then also provides a parallel pseudo-root state for the the derived classes Node, Vehicle, GustUav, and Uav. Figure 27 attributes the parallel ::running states to classes derived from SimItem. Although this is technically incorrect as in code they are actually created by the SimItem class, the figure represents the underlying principle. The Node class adds four primary states which are used to transition a Node from a bootet and running machine (standby) to an active and operational participant in the Network (operational). As a Node is also responsible for providing the communications with the network, it provides the datalink module which is integrated via the parallel state Datalink::running. This parallelism allows DatalinkInterface to operate as if it had its own state machine, yet it can access states and events of other composed or inherited classes.

The next constructed class is Vehicle. As it is a derived class and the state machine and the related states are protected and not private, it can access all the states and event transitions already constructed. The Vehicle class uses these access privileges to inject some new states in between already created ones and splits active into a piloted and an autonomous section. It qualifies the joining and leaving activities through actions that take the Vehicle to and from the route graph because, for the reasons mentioned in Section 3.2.2.1, Vehicles only operate on the route graph when autonomous.

The last state-machine-altering element in the construction chain is GustUav, whose main purpose is to implement the abstract virtual functions that Vehicle declares. To do that, GustUav communicates with the SVS instance of GUST affiliated with this Node and keeps track of the unmanned aircraft's progress through a flight plan through a simple state machine incorporated via a newly created parallel pseudo-root state GustInterface::running. GustUav also showcases one of the benefits of incorporating all the state

machines of the individual SimItems into a single one through parallel states. Events in GustInterface::running indicating that the unmanned aircraft has reached the last waypoint in a flight plan and has arrived at its commanded destination can be incorporated into the event transitions reachedGraph and reachedHome to correctly transition into their respective done states. GustUav also uses the protected status of all the states and events created so far to achieve this outcome.

3.6 Discussion

The presented simulation framework provides the foundation for research related to the utilization of multi-vehicle teams for various tasks. The framework aims to be useful to both researchers with a stronger ("developers") or weaker ("users") background in software development while trying to be close to a marketable software suite that does not cut too many corners to circumvent implementation problems which might hinder a commercial deployment but could be ignored in a purely research-driven environment. Comparable to a "design for manufacture" approach, the framework of MVS strives to support its use in a research environment through scalability in its use cases, both in numbers of Nodes and the complexity and covered area of the environment as well as through software maintainability via a modular object-oriented setup that allows source code sharing and collaboration at various access levels.

The approach to use "only" a two-dimensional environment ties in with the assumed use case of operators generating rough obstacle maps on the fly, partially relying on the added safety benefits the use of the maximum clearance route graph provides. Additionally, the related reduced computational costs should allow an easier integration into both portable devices and computers that can be used as Control Stations or avionics for smaller and smaller unmanned aircraft.

Like the localization of the route graph hopefully allows for an easy incorporation of SLAM-enabled Vehicles in the future, the object-oriented coding approach and the

separation of API-related public code and the private implementation through the use of PIMPL should enable an easier use of the MVS framework in a distributed coding environment where different research groups contribute different modules whether they are path-planning related or to implement a MANET communication protocol,or to replace the simple RF-polygon-based network simulation with a more advanced simulation engine. In unison with this modular approach, the presented framework tries to at least not categorically prevent use cases that a scenario like the motivating first responder one could warrant. At a fundamental level, choices made for one module should at least not limit options in another. Restricting the route graph to the cyclic subsets of the Voronoi diagram is one example of this: although the rest of MVS (described in Chapter 4) does not yet allow the use of hover-incapable fixed-wing, the framework-level choice to use only cyclic paths does not make it harder or impossible.

Using a geometrically-correct environment has benefits, but in turn potentially causes issues related to the fundamental problems of doing euclidean geometry in floatingpoint mathematics which are most likely of no interest to a researcher investigating in GNC improvements, for example for a tighter formation control when multiple vehicles have to cooperate in close proximity. Problems like this could be called a result of trying to stay closer to a commercially marketable product, and the presented framework certainly touches upon more of these problems: if, how, and when to simulate all seven layers of the OSI model is one example; another is the ability to provide binary compatible libraries of the core functionality of MVS through the partially cumbersome PIMPL coding idiom. As these problems cause issues for various disciplines, they not only provide an opportunity for interdisciplinary research on related issues, such as how a tight coupling of GNC algorithms with a MANET protocol could improve the performance of the networked team, but they might also enable an easier collaboration through a defined set of interdisciplinary interface parameters, e.g. codified in XSD, or the use of shared core libraries.

3.7 Possible Expansions

Touching upon many different research aspects, the presented version of the MVS simulation framework already offers various possible future additions and expansions, some of which are listed below.

Providing a Vehiclesubclass for common SVS ecosystems. The presented version of the MVS framework relies, among other third-party code, on the availability of GUST. Providing a generic class capable of transforming MVS up- and downlink data to, for example, MAVLink (the communication and messaging system used by ArduPilot) should enable using MVS with many more UAS.

Integrating COTS mapping and KML software. Instead of using a different software to generate the obstacle KML file, the GUI version of MVS could provide a picture overlay and an interface to draw obstacles directly there. This would also touch upon work required to enable dynamic environments.

Separating RF and motion maps. Currently obstacles are both blocking motion as well as the RF LOS. As particularly in urban environments many obstacles exist that are much greater hindrance for motion than for RF, such as wires, light poles, or traffic signals, then separating the motion map from the RF map could allow large safety margins around light poles which would not interfere with the provided RF polygon computation.

Improving HMI. This could involve the provision of a live updated state chart in the GUI or some way to homogenize the perceived dynamics of various unmanned aircraft in a heterogeneous Network. It also could include the generation of a dedicated researcher Control Station that has access to all data available in the simulation and a research subject Control Station that only shows information that would be available if the operator would perform a mission. The main difference would be the visibility

of Nodes not currently connected to the research subject's Control Station.

Besides this, there are the obviously larger upgrades of the framework. One example would be the integration of three-dimensional graphics so that the Control Station operator could pick an appropriate operational altitude. Additionally, the contextual information of the environment (e.g. the relation between obstacles and obstacle cells), could be used to generate a useful scene graph. Another example for a larger upgrade would be the use of a proper network simulator, for example `ns-3`, to study different MANET protocols.

CHAPTER IV

USER INTERFACE AND METHODS

Part of the motivation for developing the simulation framework described in Chapter 3 was the interest in researching a single operator interface for a team of unmanned aircraft. The prototypical application of an MVS in the motivating first responder scenario is a situation in whic a dedicated UAS operator is tasked to gather information about a fire in a high rise building.[1] In such a setup, the main problem that is to be overcome is to enable a Vehicle to operate on the far side of a building, i.e. outside the LOS of the operator or the Control Station. As the system simulated through MVS is subject to the limitations of range-limited line-of-sight datalinks (Section 3.3.2), the utilized unmanned aircraft hence need to be strategically placed in the environment to create a multi-hop communication connection (compare Figure 16(d)) to the far side of the building of interest.

This chapter introduces the user interface features and methods that MVS provides to enable a human operator to successfully overcome this Network coverage problem.

4.1 Basic Graphical User Interface Overview

Although the graphical user interface could be argued to be a part of the underlying simulation framework as its development was closely tied into the development of the framework as a debugging tool, it's design might be better explained in the context of the methods described later in this chapter. On a technical note, the GUI also is a part of the "periphery" section[2] of the underlying code and as such not a part of the "core" framework library.

Qt was chosen as the application framework as it is open source [85], platform agnostic

[1] The "Motivational Scenario" in Appendix B.1 fully describes the scenario as it was told to the participants of the related study presented in Chapter 5.

[2] The compilation of the GUI is controlled via the CMake flag `WITH_GRAPHICS`.

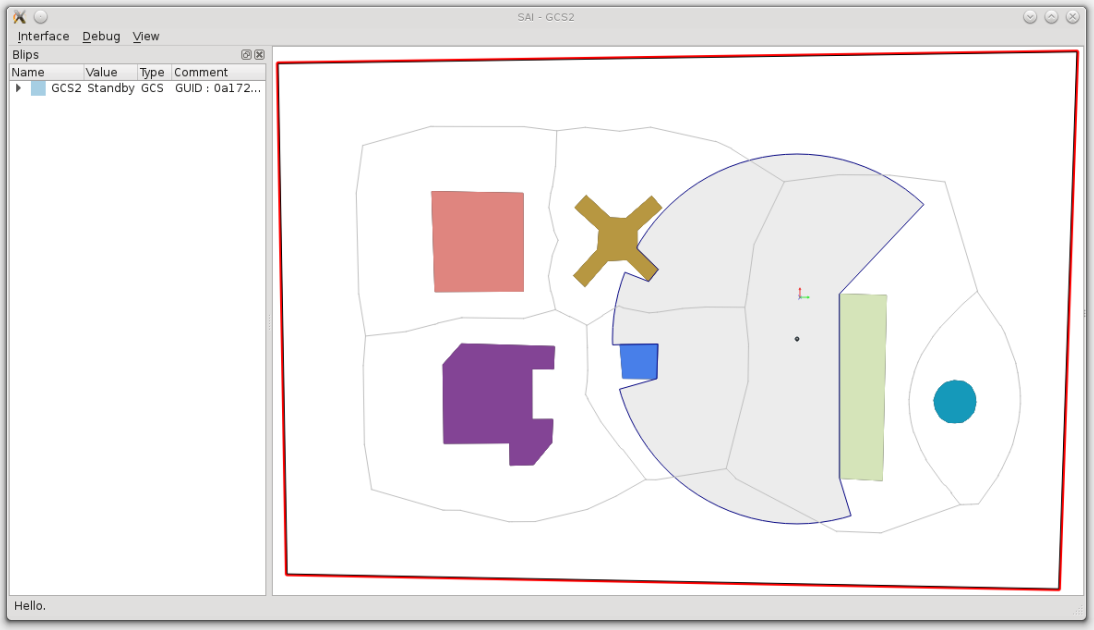


Figure 28: Basic GUI showing a Control Station Node in the training environment.

[86], and does not only provide many useful components for the core framework, but also an advanced system to build graphical user interfaces. Figure 28 shows the default interface for a Control Station that has loaded the training environment. The interface is comprised of two main parts, a data browser on the left and a map view on the right .

4.1.1 Data Browser

The data browser is a customized QtGui::QTreeView which generates a human readable representation of (parts of) the host Node's internal blip database. This database holds three types of data. Primary blip data is data that can directly be obtained from messages send out by a blip, e.g. the blip's position in three-dimensional space. Secondary data is data that can be directly computed from primary data, e.g. a two-dimensional position, and as such is somewhat redundant.[3] Tertiary data is data that is computed from purely blip

[3]Secondary data is mainly maintained to speed up data access time as it is only computed once new primary data is received and stored. Maintaining it trades memory (storing it) for computation (for recomputing it from primary data every time it is accessed) and as such can be considered a cache of sorts.

related data, i.e. primary and secondary, *and* data provided by the host. An example for tertiary data would be the RF polygon associated with a blip as it depends on the position of the blip (position) and the environment maintained by the host.[4] This data is then grouped and presented in a standard fold-out view. Figure 29(a) shows the tree view widget for the training scenario in which one Control Station and four Vehicles are active.

The data browser's ability to fold out branches can be used to adjust the level of detail of the data presented about a certain blip. The four columns of the browser provide the name of a variable, is actual value and type (which is loosely interpreted as any information that briefly qualifies the value) and a related comment. As the actual raw data kept in the host's blip database is only shown at the highest level of detail, i.e. the leaf Nodes in the shown tree, the value, type, and comment rows of aggregating groups can be used to show secondary or tertiary data. The *Position* group, for example, shows the bearing and the projected distance of the blip from the host's datum. The *Attitude* group gives a rough pointing estimation through the cardinal and intercardinal directions and the *Speed* group shows the current speed and heading of the Node.

The top or main row of a blip (the one selected and marked with the blue background in Figure 29(a)) gives the highest level of abstracted data: the blip's self reported name, its current state, its type, and a comment.[5] The states shown in the browser are not all the states a blip actually is in. MVS internally maintains a list of "reported states" which are the ones a blip reports via a dedicated State message. Every Node receiving those messages could then again select a subset to show in the browser.[6]

There obviously is an interesting trade-off to be made when selecting the number of

[4] In the current implementation of MVS the environment maintained by all hosts is identical. However, the framework in principle supports the update of obstacle data (e.g. discovering new obstacles) and as such the maintained environments cannot be assumed to be guaranteed to being identical. (See Section 3.4.4 for more details on dynamic environments.)

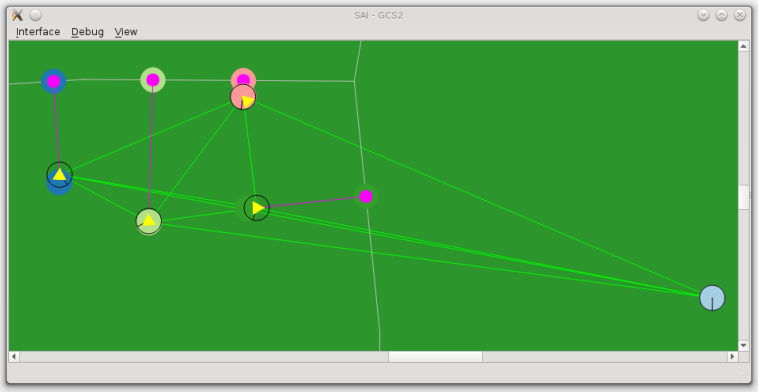
[5] The presented version of MVS shows the 128 byte globally unique identifier (GUID) associated with the blip, mainly a relict of debugging work.

[6] Presenting a comprehensive list of blip states in a single, and most likely contracted, textual row does not seem to support a great SA of the operator; Section 3.5.1 proposes a potentially much better approach for a future expansion.

Blips

Name	Value	Type	Comment
Peru3	Autonomous, Joining	Vehicle	GUID : 0a172ae1-0000-4a42-0000-0001000001c1
Position	78 m W (-100°)		relative to datum
x	-13.2574	m	North (wrt. Datum)
y	-77.168	m	East (wrt. Datum)
z	-12.3767	m	Down (wrt. Datum)
Orientation	Looking N		Body Euler Angles
φ	4.06079	°	roll
θ	-0.915413	°	pitch
ψ	8.54009	°	yaw
Speed	4.3 m/s NW (-45°)		
ẋ	0.152111	m/s	
ẏ	-0.154114	m/s	
ż	-4.2866	m/s	
Debug			Additional information for debugging.
Peru2	Autonomous, Joining	Vehicle	GUID : 0a172ae1-0000-4a30-0000-0001000001c1
Position	81 m W (-112°)		relative to datum
Orientation	Looking N		Body Euler Angles
Speed	4.5 m/s E (89°)		
Debug			Additional information for debugging.
Peru1	Autonomous, Joining	Vehicle	GUID : 0a172ae1-0000-4a1c-0000-0001000001c1
Position	98 m W (-110°)		relative to datum
Orientation	Looking NE		Body Euler Angles
Speed	4.7 m/s N (-5°)		
Debug			Additional information for debugging.
Peru0	Autonomous, Joining	Vehicle	GUID : 0a172ae1-0000-4a0a-0000-0001000001c1
Position	109 m W (-104°)		relative to datum
Orientation	Looking NW		Body Euler Angles
Speed	4.6 m/s N (2°)		
Debug			Additional information for debugging.
GCS2	Operational	GCS	GUID : 0a172ae1-0000-498a-0000-0001000001c1

(a) The data browser is a tree based viewer that presents current data form the host's blip database. Being a Qt widget, the browser can be show, hidden, detached (pictured) and re-sized.



(b) A direct connection between the pictured map view and the browser is established through the use of blip specific colors which can aid the identification process if an operator cannot use a mouse (e.g. while using a joystick).

Figure 29: The data browser provides blip specific details. Information not readily discernable from the map view, primarily the state of a blip, is presented at the highest level of abstraction and can be cross-referenced to the blips on the map view via a blip specific color.

states to be communicated or shown in the browser. On the one hand, increasing the displayed states can increase the awareness of an operator about what is happening in the system, but with an increased diversity of participating Nodes, this information can only be used in a useful way if the operator would know the related details of all the state machines involved; given the ever increasing problems related to mode confusion in complex modern day systems, properly and efficiently communicating the state of a tele-operated system is an ongoing challenge.

4.1.1.1 Blip Color

A main graphical element of that main row of a blip is a color swatch in front of the name of the blip. Every blip that is detected by a host is assigned a color to allow for an easier cross referencing and visual grouping of data related to that blip.[7] In the current implementation of MVS the ordering of Nodes in the browser represents the order of detection. The first and oldest detected Node is at the bottom, the newest detected Node is at the top. As the color assignment parallels this queue, the ownship will always be assigned the same color across all Nodes as it will always be "detected" first. Comparing, for example, Figure 28 and Figure 30(a) shows this: both Nodes are assigned the same light blue color. As the order of detection of the other Nodes depends on various external factors, the blip colors assigned to non-ownship blips cannot be assumed to be identical across Nodes.[8]

Figure 29(b) shows two cases where the blip color is used in the map view: the RF polygon center is colored solid in the blip's color, and the last waypoint in a blip's flighplan has an annular color marker around the actual magenta waypoint marker. (Another use of the blip color that is not shown is the predicted RF coverage at the last waypoint, which is

[7]The current implementation of MVS uses the Brewer's "12-class Paired" color set [8, 9], i.e. only up to twelve blips will be assigned a unique color.

[8]The choice of identifying blips by color poses various interesting HMI questions, particularly in the context of synchronizing color assignments across Nodes. The latter is particularly interesting in cases where several Control Station and human operators collaborate. If the operators have a voice communication channel, calling out colors to identify blips could speed up recognition–which would require synchronized colors. However, finding and generating sequences of n "most differentiable colors" in itself poses challenges.

outlined in the blip's color, Section 4.1.2.2.)

4.1.2 Map View

The map view in the right of Figure 28 shows the two dimensional environment as constructed by the launcher (Section 3.4.3), comprised of buildings or other obstacles (solid colored polygons) and the enclosing operational boundary as an outer limit of Vehicle movement (the red quadrilateral). The route graph through the environment is shown in light gray, indicating the maximum clearance paths in between the obstacles and the boundary. As Nodes internally use a localized North-East-Down (NED) coordinate system,[9] the datum of the host's system is also indicated through a stylized three-arrow design. A red arrow shows the x (North) direction, a green arrow the y (East) direction, and a blue cross indicates the z (Down) direction. The arrows are scaled to a length of 10 m.

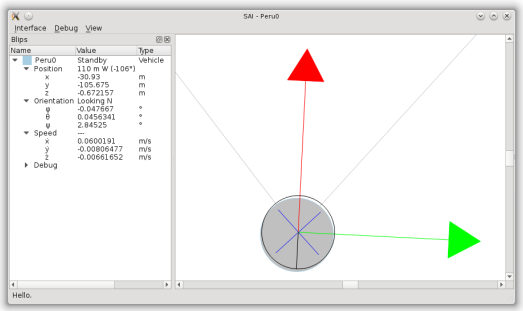
4.1.2.1 Blip Iconography

Besides those environmental features, another main feature shown in the map view is the representation of the Nodes known to the host. Figure 30 shows the representation of a blip while in standby and while moving. The (last known/communicated) location of a Node is indicated through a (black) circle with a single radial spoke indicating the "back" of the vehicle.[10] The circle is scaled to a diameter of 3 m and stylizes a helicopter's rotor disk, the spoke hints at the tail boom of a conventional helicopter. Other than a Node's location, the blip representation can also include the body carried frame (x/forward is red, y/starboard is green, z/down is blue) and an indication of the field of view of a (forward looking) camera aboard the vehicle.[11] The latter are turned of by default, but can be activated on a per-Node basis via the blip's context menu (Figures 36(b) and 36(c)).

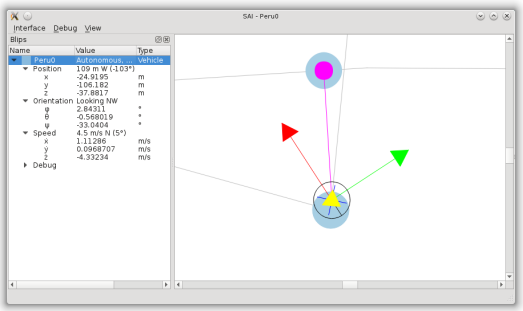
[9] MVS supports the use of geodetic, geocentric and Earth-centered, Earth-fixed coordinates through its analytical-mechanics and geometry library AMG. On a Node level, all coordinates are however kept relative to a Node-specific datum to minimize the computations related to conversions.

[10] The iconography is copied from GUST.

[11] The degree value of the field of view defaults can be set in the settings XML file for a vehicle. The presented version uses 80° for vehicles and 120° for Control Stations (the latter being a rough approximation of the binocular field of view of a human).



(a) A blip in standby. In the map view, the center marker of the RF polygon is grayed out and the blip's associated color is only indicated as a hairline outline. The data browser lists the state in the "Value" column next to the blip's name, where a color swatch indicates the (light blue) blip color.

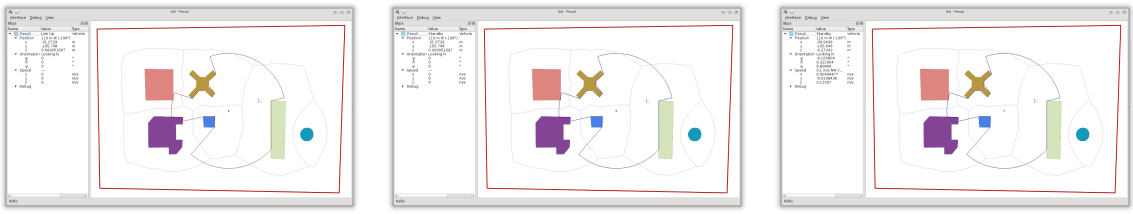


(b) A blip moving. The data browser partially lists the state "Autonomous, ..." The map view shows the colored RF polygon center marker at the position of its last update, the yellow velocity arrow indicates slow motion towards the last waypoint in the current flight plan (colored annular around the magenta marker).

Figure 30: The iconography of a blip in the map view is copied from GUST: the (black) circle stylizes a rotor disc, the (black) radial spoke a tail boom. Also shown is the body carried frame (x is forward/red, y is starboard/green, z is down/blue) and the gray field of view indicator of an aircraft mounted camera.

4.1.2.2 RF Polygon

Another important element of the blip representation in the map view is the RF polygon, which every Node in MVS computes for every other Node in the Network to build the com graph, Section 3.3.3. As mentioned there, a minimal displacement threshold is introduced to reduce the number of required RF polygon computations, depicted as the dashed circle in Figures 18(c), 19(b) and 20(b). In the map view, the threshold marker is also part of the blip iconography and it is shown as a solid circle; if the blip is in standby, the marker is gray (Figure 30(a)), if the blip is active, it is colored in the blip's color (Figure 30(b)). As a side effect of the threshold method, the black circle-and-spoke marker of the (always updated) blip position will never leave the RF polygon center marker. Figure 30(a) shows it to be fairly centered whereas Figure 30(b) depicts a case where the RF polygon update is imminent (< 1 s) as the black position marker is already close to the edge and the tip of yellow velocity arrow (showing speed in m s^{-1}) lies outside. As the computation of the field of view (FOV) of a Node's camera code internally requires the same computation as the



(a) An initialized ownship is indicated through a red RF polygon outline.

(b) When the ownship's state reaches running, the RF polygon outline changes to blue.

(c) The successful connection to a SVS instance is visible through continuous data browser updates.

Figure 31: The start-up sequence of a Vehicle is observable in the GUI and reflects the object initialization, the starting of the state machine, and the successful connection to the associated SVS instance.

RF polygon (a computation of a visibility polygon on the two-dimensional environment), is also is tied to the update threshold based on the distance to the location it was done the last time; to indicate this, the FOV indicators are also centered in the RF polygon center marker and not at the black Node position indicator. (Compare the intersection of the two gray radials in Figure 30(b) to the blue cross.)

The other half of the RF polygon is the actual indication of the covered area, which is a comparatively large graphical element in the map view. As such, it is a prime tool to communicate status information to a human operator looking at the map view as it, simply due to its size, is fairly immune to zoom level changes. The RF polygon uses both, outline and fill area, to transfer information and the color scheme depends on the combination of the status and type of the host (i.e. the Node hosting/generation the GUI) and the blip presented. In the presented implementation of MVS this leads to four main perspectives:

- A Vehicle host depicting the ownship,
- a Vehicle host depicting another blip,
- a Control Station host depicting the ownship,
- or a Control Station host depicting another blip.

The graphical representation of the outline of the ownship is the least ambiguous: the

ownship always has a (bolder) outline to allow for an easy "this is me" identification and uses two colors: a darker blue for the nominal running state and red for any non-nominal situation. Figure 31 shows the use of the outline for the ownship during the start up sequence of a Node.[12] The map view shows the ownship as soon is it's object is constructed and initialized in code, but as the Node's state machine is not running yet,[13] the outline is red (Figure 31(a)). Once the state machine is started and the Node is in the "running" state, the outline of the RF polygon turns dark blue (Figure 31(b)).

The outline used for other blips, i.e. not the ownship, depends on whether the host is connected to a Network or not. Figures 32 and 33 show how a Vehicle depicts other blips while a Control Station is creating a network. A discriminating factor is the Network status of the host, i.e. whether the blip and the ownship are members of the same Network or not. If the host is not part of a Network (i.e. in standby), other blips in standby are shown with a gray outline (e.g. Figure 32(a)) whereas blips that are active (i.e. part of a network, but not relevant to the host) are given a green outline (Figure 32(b)). Once a host joins a network, which is synonymous to reaching the active state, the use of outlines changes as an active host generally does not draw outlines around blips. This switch can easily be seen when comparing Figure 32, in which the hosting Node is the first vehicle joining, and Figure 33, in which it is the last one. Both sequences start identically; the first picture shows all Nodes in standby, the second one depicts the Control Station going active, which "creates" the Network (compare Section 4.2.1). In the sequence shown in Figure 32 the hosting Vehicle is then the next blip to join the network, and all following pictures show the RF polygons filled and without an outline. In the sequence shown in Figure 33, the hosting Vehicle is only joint in the last step, keeping the RF polygons outlined and not-filled up until then.

[12] The ownship in this case is a vehicle, "Peru0." Figure 28 shows a situation identical to Figure 31(b), but for a Control Station, "GCS2."

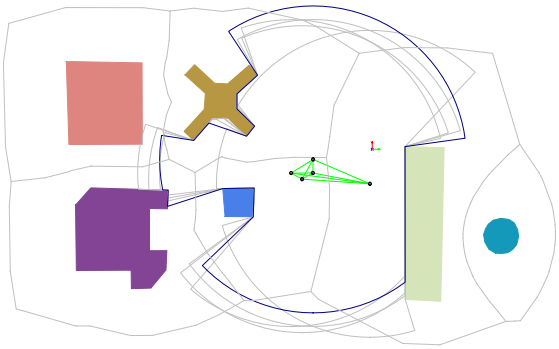
[13] The XML start up script (Section 3.4.3) allows the setting of a delayed start of the state machine. This allows a slight desynchronization of Nodes to achieve a more homogeneous use of network and communication bandwidth.

As Control Stations are the seed Nodes to create a network, Control Stations are considered to always be part of a Network (the one they could create by *joining*), no matter whether they are active or still in standby. As such the map view of a Control Station will never show outlined RF polygons as they are always shaded.[14] To indicate the difference state, the RF polygon of a Control Station in standby is filled gray and only switched to green once the Control Station goes active. Figure 37 shows an abbreviated version of the sequence shown in Figure 32 drawn by the Control Station creating the network.

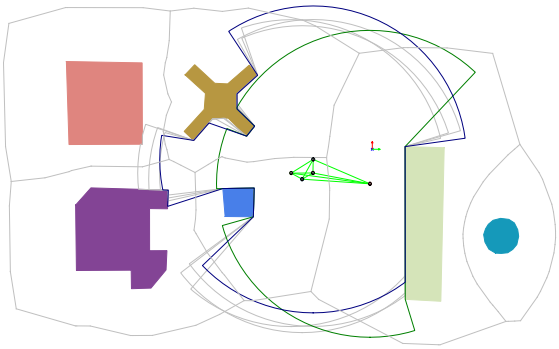
The shading or filling of RF polygons is generally driven by the simple goal to highlight all "allowed" connected areas. The guiding idea is that wherever there is a green shade, a blip can go. As such, the graphical representation of the inner area of a RF polygon is fairly simple: a green shade when the related blip is part of the host's Network and the area is allowable and a gray shade for the RF polygon of blips that could potentially be joining the hosts network. However, there are small differences in what is the "allowable" area, depending on Node type.

Figure 34 showcases the most common situations by providing the the representation of a common Network setup by all Nodes involved. For a Vehicle the "allowable" area is wherever the host has connectivity to its Control Station–assuming all other Nodes remain stationary. The area shaded green is identical to the RF polygons of all Nodes which are in the connected component of the Control Station *if* the host Vehicle were to be disconnected from the com graph (compare Figure 18, letting the star icon represent a Control Station). The first blip shown in Figure 34(b) is not a separating vertex in the com graph (disconnecting it from the graph does not affect the connectivity of the other Nodes) and as such the RF polygons of all other Nodes are shaded green; the same is true for the fourth blip (Figure 34(e)) as it is the last vertex in a com graph branch. The second and third blip, however, do separate the graph when they are disconnected. Hence both only shade the

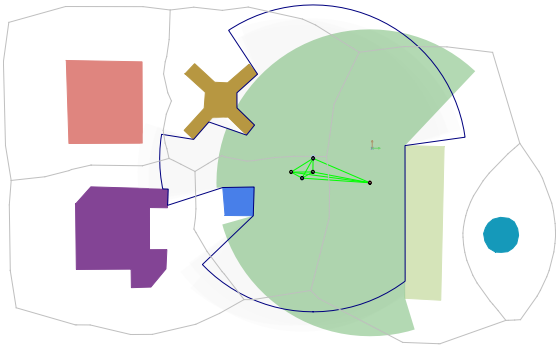
[14] RF predictions are drawn outlined in Control Station map views, but they are conceptually different from RF polygons, Section 4.1.2.4.



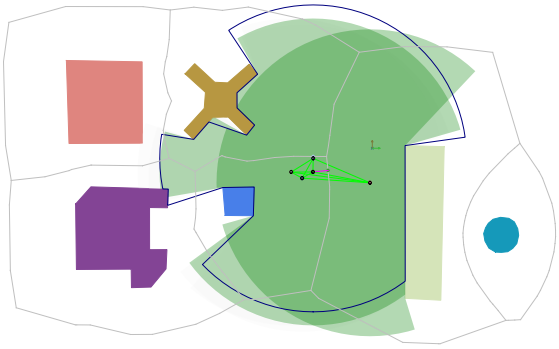
(a) The host Vehicle is running (ownship outline is blue) and it and all other Nodes are in standby (outlines are gray).



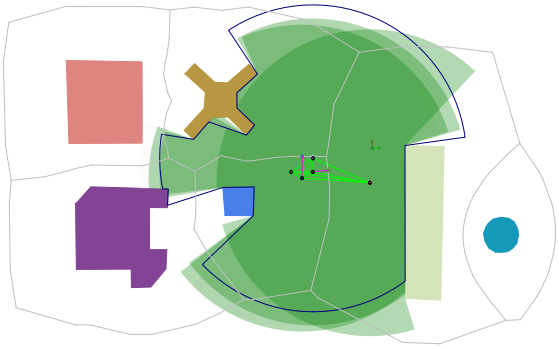
(b) A blip has gone active, i.e. joined a network; its outline is green.



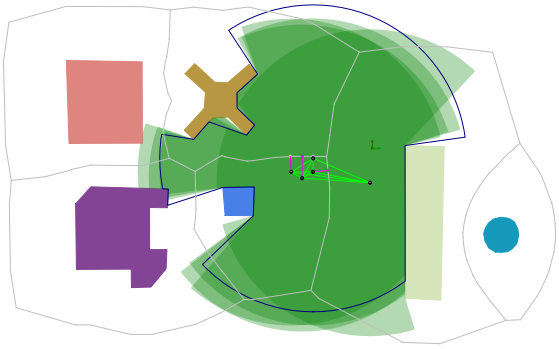
(c) The host has joined a Network (RF polygons are filled); the area com-connected to the Control Station of the host is shaded green.



(d) Another blip has joined the network, its RF polygon shade changed from light gray to green.

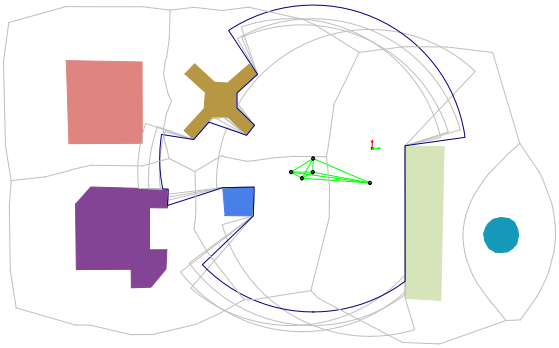


(e) A fourth blip has joined the network. The darker green an area is shaded, the more Nodes can communicate with it.

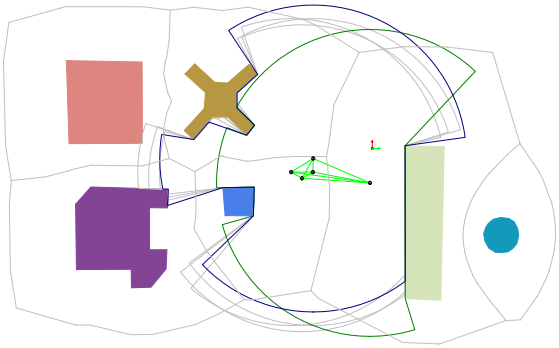


(f) All blips are joined. As long as the host stays within the green area, it is com-connected to its Control Station.

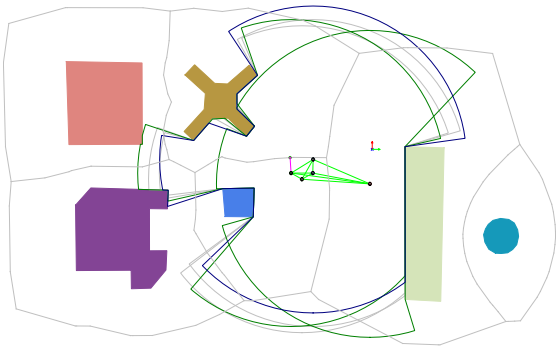
Figure 32: A Network formation sequence as depicted by the first vehicle that is joint to the network. The hosting Vehicle is joint to the Network after the second step as from the third picture on the RF polygons are filled and not any longer outlined.



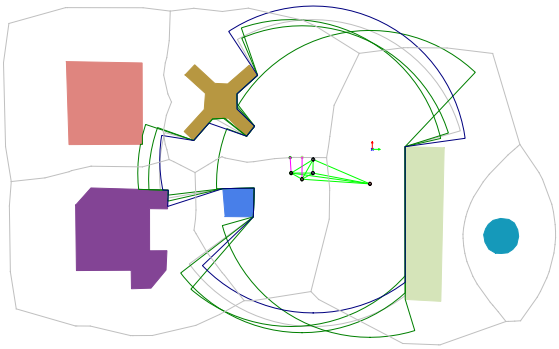
(a) The host Vehicle is running (ownship outline is blue), all other Nodes are in standby (outlines are gray).



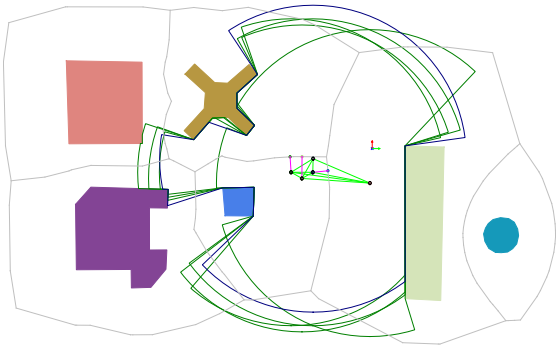
(b) One blip joined a network.



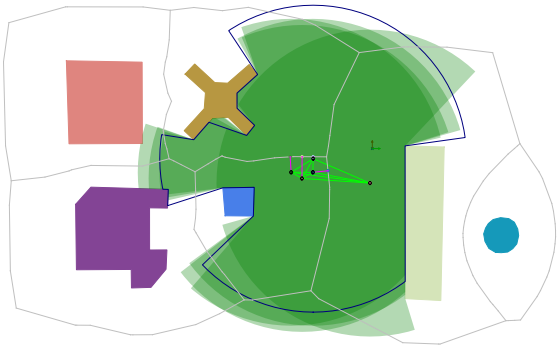
(c) A second one did.



(d) A third one joined.

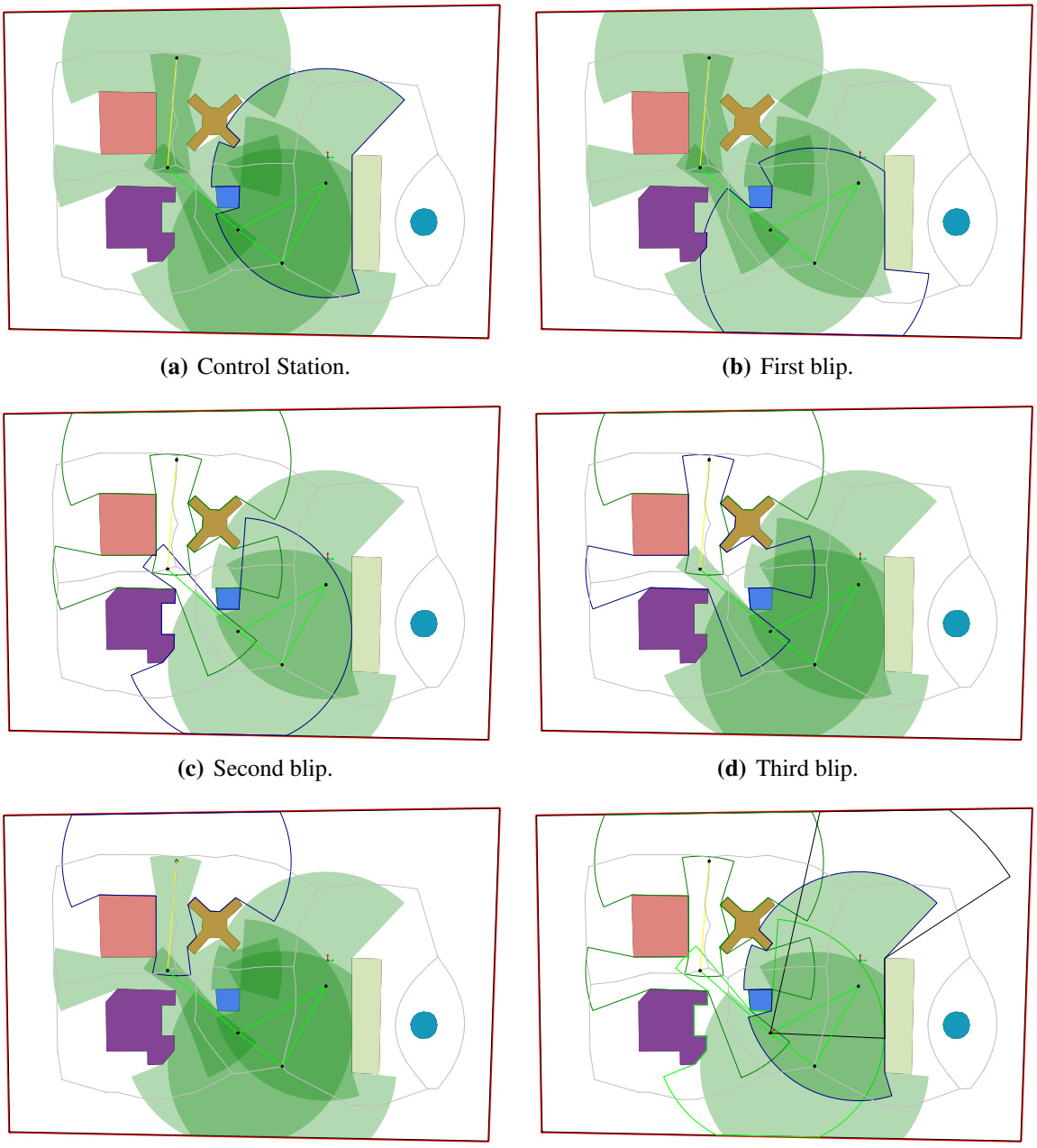


(e) And a fourth one.



(f) The change from outlined to filled RF polygon indicates that the host Vehicle has is now joint.

Figure 33: A Network formation sequence as depicted by the last vehicle that is joint to the network. The hosting Vehicle is joint to the Network after the fifth step as only in the last picture the RF polygons are filled and not outlined any longer.



(a) Control Station.

(b) First blip.

(c) Second blip.

(d) Third blip.

(e) Fourth blip.

(f) Joystick control of second blip.

Figure 34: Pictures (a) through (e) show the identical Network situation, depicted from different Nodes. The "allowable" area (shaded green) is interpreted slightly different: for a Control Station (a) all covered area is "allowable," for a blip (b–e) only the area that provides a connection to the Control Station is allowable. When a Control Station engages joystick control for a blip (f), the "allowable" interpretation changes to the same one a blip is using: only areas connected to the Control Station are shaded green (compare (c) and (f)).

connected component containing the Control Station green (Figures 34(c) and 34(d)).

Figure 34 also shows how the outlines of RF polygons of blips downstream of the Control Station are using the same green that has been previously introduced on page 92 to to indicate "active, but not relevant." The figure also shows that a Control Station marks a joystick controlled blip through a bright green outline.[15]

4.1.2.3 Flight plans and Waypoints

Flight plans, at this point a simple collection of waypoints, are used by Vehicles to plan and execute their motion. Providing this feature is one of the few requirements MVS has for a SVS instance to be attachable to a Node (compare Section 3.1) and as such a unified graphical representation can be achieved across all possibly used SVS systems.

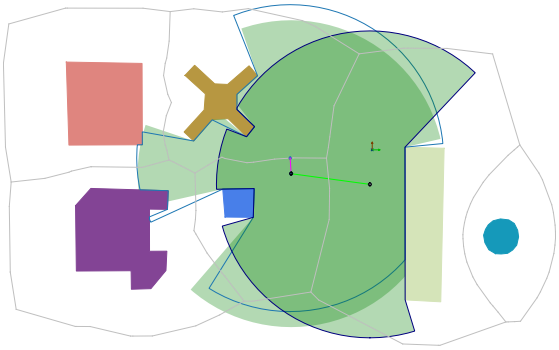
Sticking with a somewhat industry standard, waypoints and flight plans in MVS are colored magenta. Figure 35 depicts how a Control Station and a Vehicle represent flight plans and waypoints slightly differently, depending on whether a flight plan is shown for the ownship or another blip.

Flight plans and waypoints are always shown in their entirety for the ownship. The right column of Figure 35 shows this for the case of a Vehicle. Upcoming waypoints are represented as magenta dots, connected by a magenta line; already passed waypoints are darker and not connected. The last waypoint in a flight plan, i.e. the destination, has an annular marker in the color of the blip.

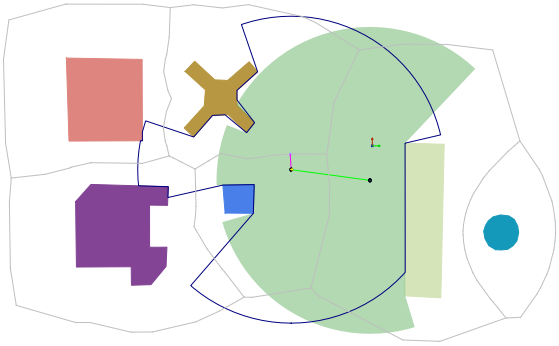
This marker is important in so far as that non-ownship flight plans are truncated after the third upcoming waypoint.[16] The left column of Figure 35 highlights this at the example of a Control Station. Truncation seemed necessary as particularly in multi-vehicle operations the crisscrossing and mostly overlaying complete flight plans did not only not help but potentially even reduced understanding of which Vehicle was moving where. The annular

[15] Joystick controlled blip's are also indicated by showing the FOV indicator as well as the body carried frame, see Figure 34(f).

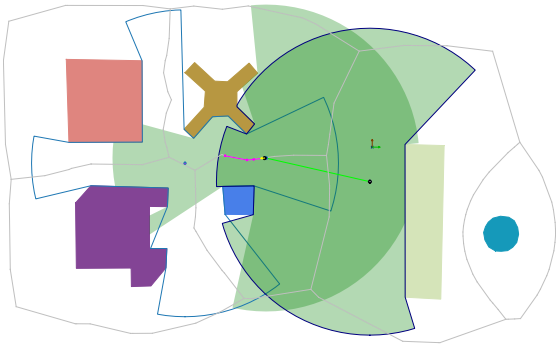
[16] The number of upcoming and passed waypoints shown for non-ownship flight plans be easily adjusted in code.



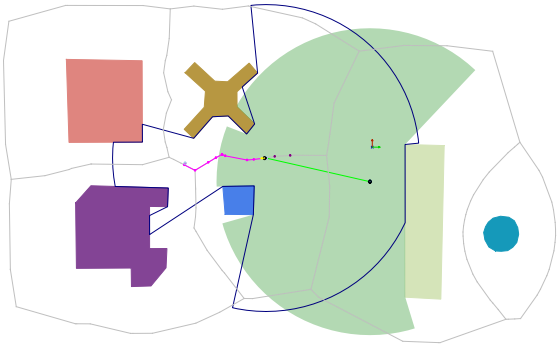
(a) Control Station joins a blip to its network; last waypoint and RF coverage prediction are shown.



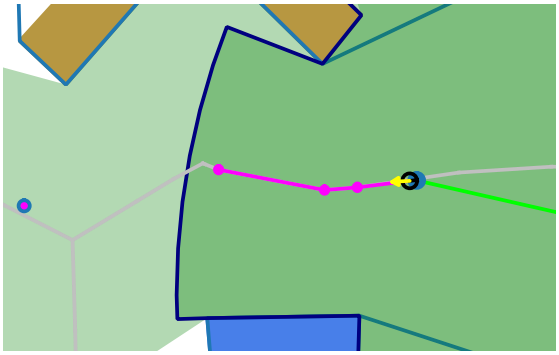
(b) The blip also highlights its last waypoint but does not show a coverage prediction.



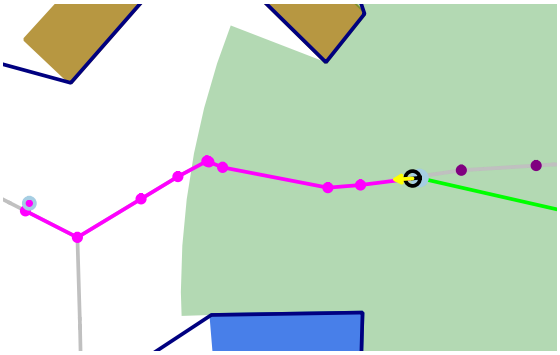
(c) The Control Station shows an abbreviated flight plan and the RF prediction at the destination.



(d) The ownship flight plan is shown completely by blips, including passed waypoints.



(e) Non-ownship flight plans are truncated. The destination waypoint is shown with a colored annular marker.



(f) The ownship flight plan is shown in its entirety, including passed waypoints (darker dots).

Figure 35: The left column shows a Control Station's map view representation of joining a blip to its Network and commanding it to a location. The right column shows the same scenario, represented by the commandeered blip.

marker allows to quickly see where a particular blip is moving towards, even when there is no continuous magenta line connection of the blip to its destination. As the Vehicle are bound to move on the route graph when not under joystick control, a human operator should normally be able to guess the path a particular blip is taking correctly. The truncated flight plan can help when a blip is close to an intersection by indicating which way the blip is going to turn.

The visualized flight plan is by no means an accurate prediction of the path a particular Vehicle will take, it solemnly represents the flight plan that the SVS interface in the Vehicle-derived class (e.g. GustUav, Section 3.4.2) sent to the SVS instance.[17]

4.1.2.4 RF Coverage Prediction

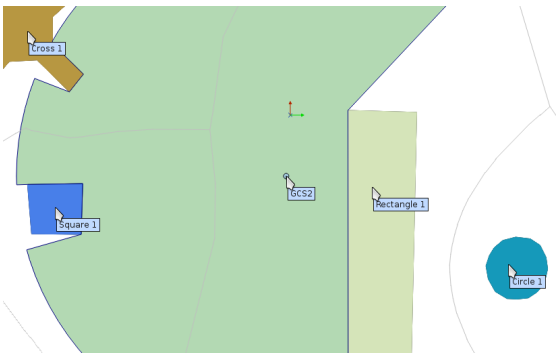
To enhance the understanding and prediction of the Network's coverage area, the last waypoint markers of flight plans are also enhanced with a "prediction" of the coverage provided by a Node placed at this location. MVS effectively calculates a RF polygon for that location which is then shown in the map view as an outlined, colored in the blip color of the Vehicle that is headed to that location. Figure 35 shows these RF predictions in the left column pictures as the light blue outlined RF polygons.

The RF coverage prediction polygons are also computed when a blip is placed, for example through drag and drop (Section 4.2.2), which allows the Control Station operator to predict the covered area and better dispatch blips.

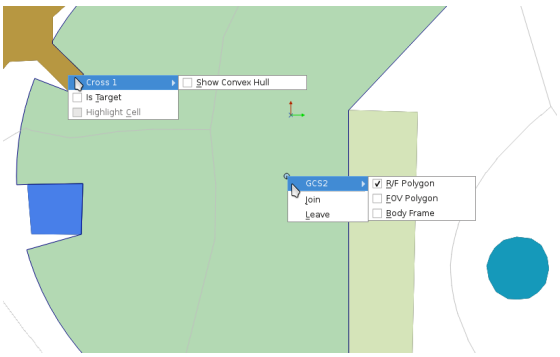
4.2 Command Interface

The command interface available to an operator (or researcher) is limited to capabilities built into the GUI interface. Compiled console versions of MVS consequently do not

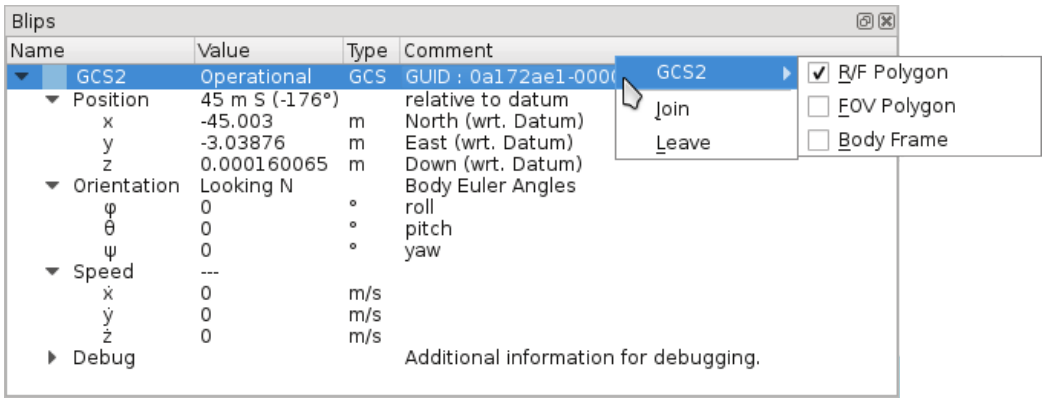
[17] The shown line is particularly *not* comparable to the "magenta line" that GUST shows in its native scene viewers as the latter is an onboard created prediction of the actual trajectory which includes the vehicles dynamics. MVS does *not* do something comparable.



(a) Tooltips help to identify various elements in the environment.



(b) Context menus give access to element specific options.



(c) The browser also provides access to the Blip context menu.

Figure 36: Composite images show options for a mouse-based interaction with the elements in the environment. Tooltips identify the elements (obstacles and the Control Station) and "right-click" context menus allow interactions. The context menus also identify the clicked object and give options to modify the graphical appearance in a fold out sub-menu.

provide any capabilities to directly interact with the running (console) application.[18]

Figure 36 shows the main components of this mouse-based interface, which is centered around the use of tooltips and context menus commonly found in GUI interfaces. The menus are created in a similar fashion for all elements in the environment, i.e. both blips as well as obstacles show an identifier as the first row (identical to the name shown in the tooltip of that element) which can be expanded to get access to graphical options for the element at hand. Below this line are commands applicable for the element, with currently

[18]This limitation includes the shutdown of a Node, i.e. a started console application has to be killed by other/external means.

unavailable options grayed out.[19] The order and grouping of the commands follows the inheritance hierarchy of the element, i.e. a blip would list Node related commands higher than Vehicle related commands. Very specific commands, mainly useful for debugging, can additionally be found in the dedicated "Debug" menu in the menu bar of the main MVS window.

4.2.1 Creating Networks

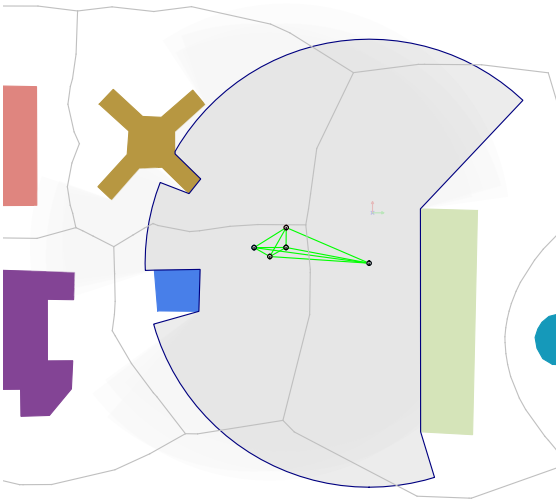
An initial requirement in all MVS scenarios is the creation of a *MVS network*, which describes a group of Nodes working together. Only a Control Station can create a Network; it does this by *joining* Nodes to it, starting with itself.

Figure 37 depicts how a Control Station creates a Network consisting of a total of five Nodes: one Control Station, itself, and four Vehicles. The Control Station starts from standby and joins itself to the newly created MVS network. The operator performing this step can do this via the Join command in the ownship's context menu. After this has been done, issuing the Join command to other blips in standby affiliates them with the network. A Control Station can only issue the command after itself has joined, which is why the first step in creating a Network is joining the Control Station itself.

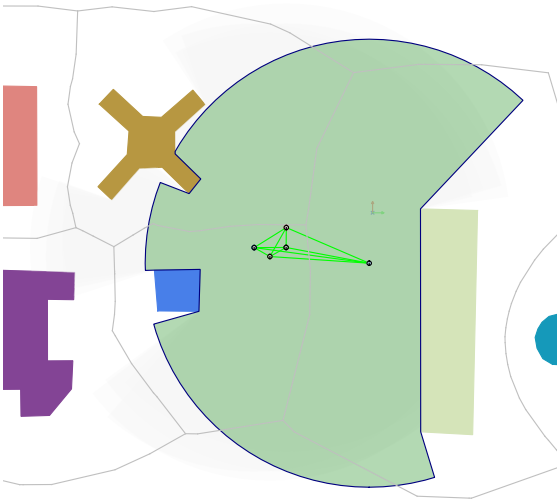
The default behavior of a joining Vehicle is to move to the closest position on the route graph at a predetermined operational altitude (movingToRouteGraph in Figure 25).

This admittedly rather cumbersome first step of joining the Control Station is introduced to maintain the ability to have several Control Stations active at the same time, potentially even operating in the same Network as an underlying idea for a future setup of MVS would be that a Network could be seen as representing a separate virtual local area network (VLAN) inside the MANET spanned by all Nodes, i.e. that several groups of Nodes can collaborate in the same shared space and environment. This scenario also foresees that Vehicles are only joined to a Network while they are needed (e.g. because they

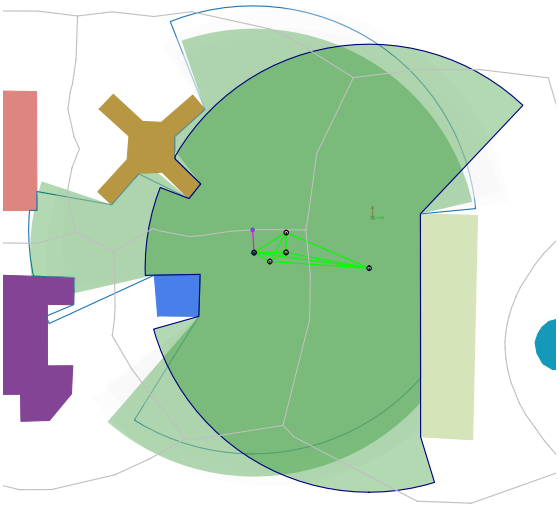
[19] Figure 36(b) shows the "Highlight Cell" command grayed out as this requires a global option "Obstacle Cells", settable via the main window's "View" menu, to be set first.



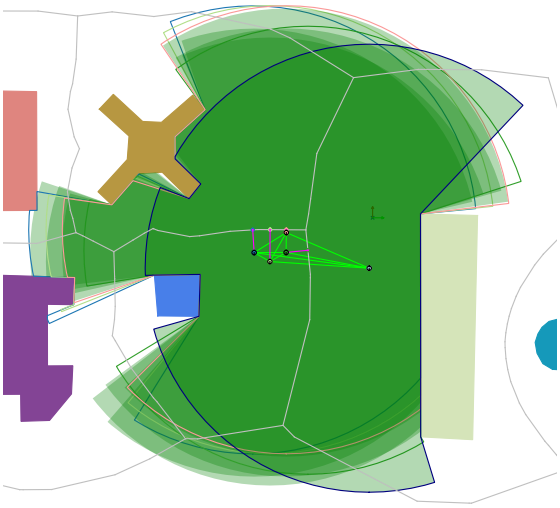
(a) All Nodes in the Network are in standby. The ownship's RF polygon is outlined in dark blue. As all Nodes are within range, the green com graph is fully connected.



(b) The ownship is operational. The RF polygon's center marker is colored in a Node-unique color and the network's coverage area is highlighted in green.



(c) The first blip is joining the network. The current RF coverage is shaded green, the predicted coverage at the destination outlined in the blip's color (a lighter blue).



(d) The remaining blips are also joining. Darker shades of green indicate more Nodes provide coverage for that area.

Figure 37: A Control Station creates a Network through joining itself and other blips to it. The joint coverage of the network is represented in shaded green, showing the area that is (multi-hop) reachable.

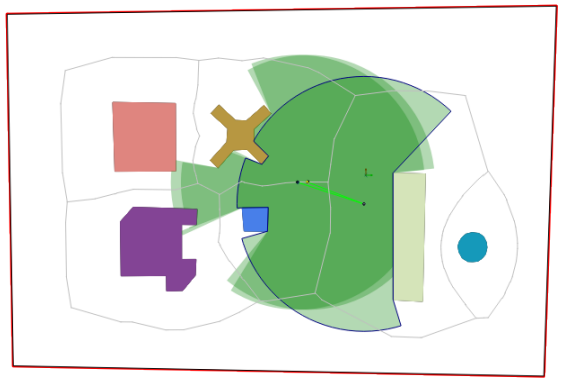
carry a specialized payload) and shared among the different Networks. During the initial beta tests of the GUI as well as in the post-experiment questionnaire of the human subject study (Chapter 5), many participants voiced a wish for a Join All function which explicitly is not provided to allow a forward compatibility to such a larger usage scenario.

Disbanding a Network works in reverse of the creation process: instead of the Join-command the Control Station issues a Leave command to the blip. If the complete Network is to be dissolved, the Control Station can Leave itself which will automatically trigger a Leave event in all associated blips. The default behavior for leaving blips is to return to their start up position (movingToHome in Figure 25).[20]

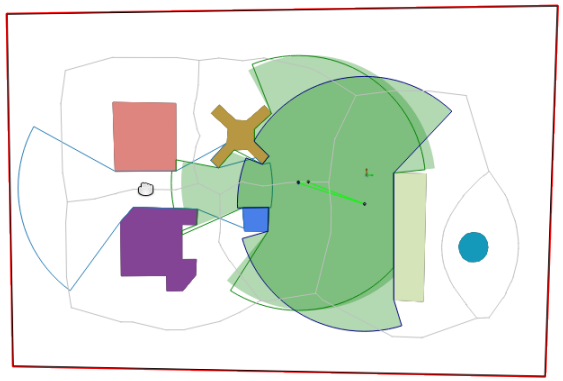
4.2.2 Moving Blips via Drag and Drop

As no direct or console-based command interface is provided, the only way to command a Vehicle to a certain location is via a drag and drop interface in the map view part of the main application window.

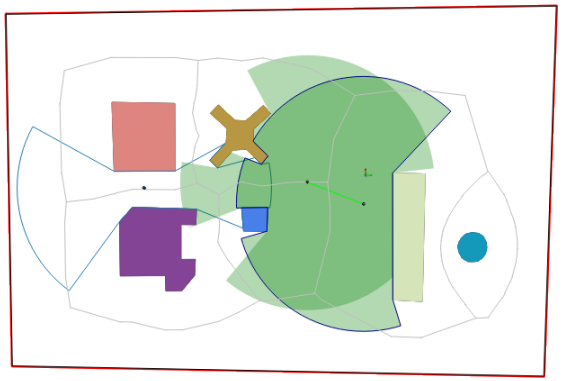
[20] In order to allow for GUST-based Vehicles to be rejoined again, movingToHome actually commands them to a position 10 m above their start up location to avoid (simulated) interference with the ground as the 0 m above ground level (AGL) altitude reference plane might not be perfectly synchronized between MVS and all GUST instances. In GUST this could be overcome by commanding an auto land at the end of the movingToHome action.



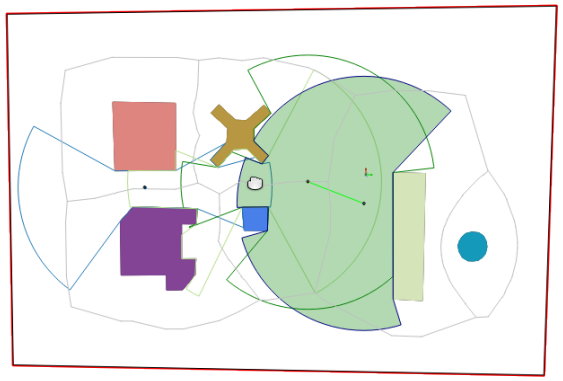
(a) Initial Network: two Vehicles and a Control Station.



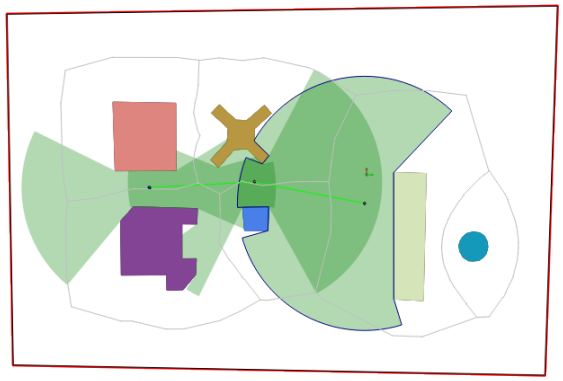
(b) A Vehicle is send outside the covered area.



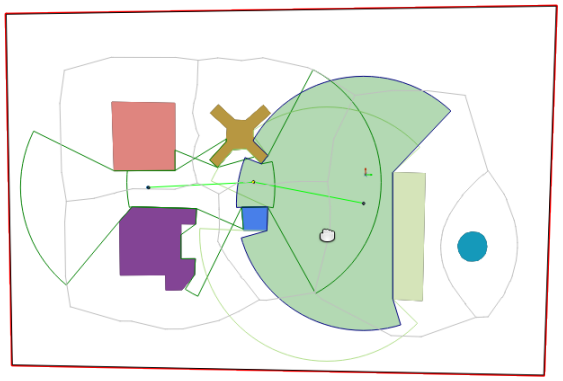
(c) Without a connection, only the RF coverage prediction is shown.



(d) The second Vehicle is commanded into the intersection region.



(e) The communications connection to the first Vehicle is recovered.



(f) Dragging the relay reveals the collapsing Network coverage.

Figure 38: MVS provides a simple drag and drop interface to command blips to new locations. The operator of a Control Station can simply point, click, and drag a Vehicle to the commanded position. While dragging, the green shaded Network coverage area is altered to highlight the areas the currently dragged blip will have connection to the Control Station. The operator is not limited to the connected green area and can drop a blip at any (permissible) location.

Figure 38 depicts the process of moving two blips from their initial position on the route graph to locations further away from the Control Station. In the first step (Figure 38(b)), a Vehicle is send outside of the RF coverage area of the Network (the "drop" location is outside the green shaded area), but that is no problem as long as the commanded Vehicle is connected to the Control Station at the time the command is send (i.e. the "drop" event). The only non-permissible areas to send Vehicles to are locations inside obstacles or outside the operational boundary, i.e. the permissible area is the interior of the environmental polygon (Section 3.2.1. The hand shaped cursor will change to a "forbidden" sign if a blip should be dragged outside the permissible area.) The next command (Figure 38(d)) steers the second Vehicle into the intersection region of the Control Station RF polygon and the predicted coverage of the Vehicle send away in the first step[21] and reestablishes the communication connection of the Network.

While the operator is dragging a blip, MVS continuously updates and draws the predicted RF coverage area (Section 4.1.2.4) for the current cursor location, shown, for example, in Figure 38(b). As the RF coverage prediction is also drawn for the last waypoints of flight plans, this allows an operator to plan the placement of blips to achieve coverage of a certain area and to ensure that Network connectivity is maintained, Figure 38(d).

As capturing a blip with the mouse cursor can be tricky at times, particularly when the operator is pressed for time or the map view is zoomed out or panned to a different region, the drag and drop interface also works when it is initiated from the data browser: an operator can click-and-hold the primary row of a blip (the one listing the name and showing the blip's color, Figure 36(c)) and simply drag it into the map view part. This method is particularly helpful in situations where blips are clustered or when it does not matter where a blip currently is.

[21] As the Vehicle is outside the RF coverage of the Network, the Control Station (and every other Node for that matter) should not be able to receive the position messages of that vehicle. As outlined in Section 3.3.3, pages 60ff., MVS currently still displays these Nodes under the argument of a very accurate estimation. A consequence of this is that Figure 38(d) outlines the blip's RF polygon in the blip's color, marking it an estimation or prediction, whereas Figure 38(f) shows it in green, marking it actually "known" information.

4.2.3 Direct Joystick Control

Beside the waypoint and flightpath interface, MVS also requires that a SVS provides an interface to allow some sort of direct "piloting" (Section 3.1) to allow an operator to directly control a Vehicle with a joystick as an alternative to the much less precise mouse-based drag and drop method. Doing so gives the joystick controlled vehicle the role of the primary unmanned aircraft.

As all SVS integrated into MVS are assumed to provide basic GNC capabilities, "piloting" a Vehicle in MVS is rather comparable to "telling the autopilot what to do" instead of traditional R/C piloting.[22] As a consequence, the joystick axis are not mapped to traditional inputs like aileron, elevator, rudder, *etc.*, but they are mapped directly to velocities (forward, backward, (strafing) left, (strafing) right) and rates (turning left, turning right). In order to allow a multitude of SVS instance to be operable in the MVS framework, the individual direct control methods have to be unified, adaptable to various joysticks, and potentially need to allow for different preferences with respect to the so called *Mode* of a transmitter like joystick.

This unification is however fundamentally limited by some constraints given through the type of unmanned aircraft used as a Vehicle. Rotary wing aircraft, or generally all hover capable aircraft, can use all of the previously mentioned velocity commands. Giving no command then directly translates to no velocity, i.e. a stationary hover. Aircraft incapable of a hover, for example most fixed wing aircraft, can obviously not achieve a zero-velocity state and need a constant (forward) speed. In this case the previously mentioned direct joystick commands need to be adapted. One possible example would be to replace the forward/backward velocity command with an accelerate/decelerate acceleration command and implement a minimal speed. Additionally the left/right strafing velocities do

[22] Observing the participants of the human subject study (Chapter 5) during the initial training and familiarization phase seemed to indicate that the more experience a participant had with traditional (R/C) controls, the longer it took them to adjust to the given MVS joystick interface.

not make sense for fixed wing aircraft as they, in two dimensions, essentially represent Dubins vehicles. Although the underlying framework supports hover incapable unmanned aircraft through the provision of the cyclic route graph, the presented implementation of MVS assumes that all Vehicle are hover capable.

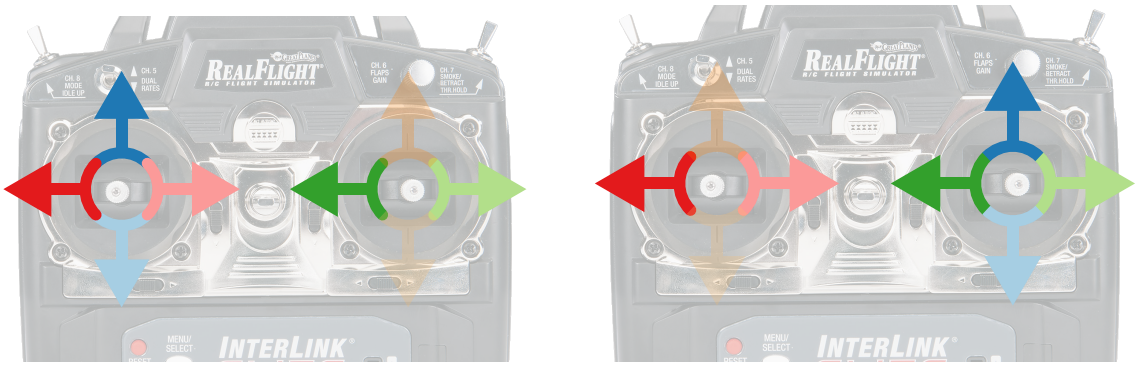
The adaptation and calibration to different joysticks can be taken care of in the setup configuration files of any Node that provides an HMI module in the form of an HumanMachineInterface SimItem. To do so, the XML setup file provides maximum and minimum values so that all the raw values read from the controller[23] can be properly normalized and assigned to the related commands. The *Mode* mapping is taken care of in the assignment of the joystick axis to the traditional inputs (rudder/yaw, elevator/pitch, throttle, aileron/roll), which then in turn are assigned to the velocity and rate commands: the elevator/pitch input is used as forward and backward, aileron/roll as left and right strafing, and rudder/yaw as turning left and right. The throttle input is technically mapped to an up and down motion, but as all movement is restricted to happen at a predefined operational altitude (governed by the altitude for which the utilized environment is valid), it's effects are negated through a code internal altitude hold mode.[24] Figure 39 depicts the differences between the two most common gimbal setups found in RC transmitters, the so called *Mode 1* and *Mode 2* configuration.[25]

In addition to the gimbal inputs, the joystick input also defines a "dash" functionality on a button. "Dashing" provides a constant bias in forward velocity and allows a joystick controlled Vehicle to move much faster than the default traversing speeds of autonomous Vehicles, i.e. it enables an operator to use higher speeds in open areas (piloted Vehicles are not restricted to stay on the route graph) while maintaining higher control when operating close to obstacles. As the operator only commands an onboard autopilot of an SVS

[23]MVS uses the Simple DirectMedia Layer [60] library to access joysticks.

[24]The altitude hold mode is tied to a switch on the joystick. However, toggling the altitude hold was mainly a debugging necessity and participants in the human subject study were not told how to disengage it.

[25]For the human subject study joysticks in both configurations were available to the participants.



(a) A *Mode 1* setup has pitch (blue) and yaw (red) on the left stick and roll (green) and throttle on the right stick.

(b) A *Mode 2* setup has yaw (red) and throttle on the left stick and pitch (blue) and roll (green) on the right stick.

Figure 39: The joystick input mapping assigns the commanded motions to the stick axis. Forward and backward is indicated in blue, (strafing) left and right in green, and turning left and right in red. Up and down motions were not assigned as all movement was confined to a specified operational altitude. (Controller image © 1997-2015 Great Planes Model Mfg. A subsidiary of Hobbico, Inc. Used with permission.)

instance, simply letting go of the sticks brings a hover-capable vehicle to a stop. This functionality can be helpful in cases where the inherent delay of using velocity and rate commands with actual vehicle dynamics leads to cases of pilot induced oscillation (PIO).[26]

As joystick control is another way for the operator to get a Vehicle to a location, leaving the joystickControl state has the same effect as being done with with executingExternalCommands: the Vehicle's sate machine transitions into pilotedLoiter (compare Figure 25), i.e. the Vehicle stays where it is.

4.2.3.1 Single Operator Support

A main underlying driver for the development of MVS was the single-operator multi-vehicle scenario in which a single operator can command and control a complete team of unmanned aircraft. This works well on a higher level, for example through using the drag and drop interface, but when an operator takes direct joystick control of a Vehicle, the operator's focus is presumably on that vehicle–and most likely on that vehicle alone–which

[26] Simply letting go of the sticks and letting the onboard GNC algorithms stop the oscillations was shown to the participants of the human subject study to be a fast and easy way to stop "unwanted" behavior of a piloted Vehicle.

warranted the differing roles of the single primary unmanned aircraft

To support an operator while joystick controlling a Vehicle, the Control Station's map view changes which area is shaded green. As soon as a Vehicle is put into joystick control only areas that allow a communication connection between the primary unmanned aircraft and the Control Station are "allowable" and shaded green. Figures 34(a) and 34(f) highlight the difference. In the first picture, no Vehicle is under joystick control and the complete coverage provided by the Network is shaded green. In the second picture, however, one blip is under joystick control (the one with the shown FOV indicator). Only the coverage provided by the connected component containing the Control Station (compare Figure 18(d)) is shaded green as leaving this area would terminate the communication connection between the joystick controlled Vehicle and the controlling Control Station.

4.3 Aiding Methods

The basic command interface through drag and drop and direct joystick control allows an operator to position Vehicles throughout the environment and build RF coverage around the building of interest. The methods described in this section expand upon those basic capabilities in order to improve the overall system usability and performance.

4.3.1 Lost Link Procedure

Both control methods, drag and drop and joystick control, allow the operator to move a blip out of the (current) coverage of the Network. Maneuvering a Vehicle out of range while it is under direct joystick control is most likely unintentional and a "feature" of the underlying implementation of the joystick message handling. A GUST SVS instance, for example, uses an internal timer to check for a connection to its GCS, replicated through a GustUav in the MVS setup (Figure 23). As GustUav does not explicitly send a zero message upon loosing connection to its controlling Control Station (it simply does not send new messages), GUST treats this as a "bad link" situation and continues with the last received joystick input for a few seconds before it stops. This leads to a situation where it is

possible to "drift" out of range of the Control Station through maintaining the established velocity at the time the communication to the Control Station was lost.[27]

Moving a Vehicle out of range via drag and drop can, however, indeed be intentional and helpful as Figure 38 shows. In the depicted process, the operator first sends a Vehicle to a far away out of range location and then sends a second Vehicle to rebuild the communication direction. At the same time loosing link to a dragged and dropped Vehicle can also be unintentional, especially if the operator gets confused about the area covered by the Network. This can, for example, happen when one blip is dragged while another Vehicle is under joystick control as in that case, shown, for example, in Figure 34(f), the currently covered area is not identical to the green shaded area and the operator would have to utilize the com graph to determine which blips indeed provide coverage where.[28] As a consequence of any of these cases, loosing link to a Vehicle can easily happen. One possible way to mitigate this situation would be to use the exact same process used in the drag and drop example (Figure 38), i.e. the operator could use other Vehicles to rebuild the link to the stray blip. This, however, assumes that there are Vehicles left to be repositioned and that the operator notices the situation.

To mitigate the effect of a lost link in cases where there are no other Vehicles available to "rescue" the lost one or the operator either has not noticed or does not have the time to deal with the situation, every Vehicle employs a rudimentary return to launch site (RTLS) procedure in case of a prolonged lost link situation. The currently implemented process is triggered when a Vehicle has been disconnected from its Control Station for more than 120 s. After that the Vehicle automatically starts a flight plan back to the last known location of the Control Station. Assuming that the Control Station moves relatively slow (if at all), this should eventually bring the Vehicle back into the communication range of the Network,

[27] If the "dash mode" is active when the communication is lost, the a GustUav can actually traverse a fair amount of distance before being commanded into a loiter. This behavior, although not in its extremes, is actually intentional as it allows an operator to "drift" through small coverage gaps.

[28] Getting confused about the coverage area is a failure case repeatedly observed in the human subject study presented in Chapter 5.

which will trigger a stop of the Vehicle and a transition into loitering.

Figure 40 shows how the RTLS process plays out in the map views of a Control Station (left column) and the returning blip (right column).

4.3.2 Obstacle Coverage

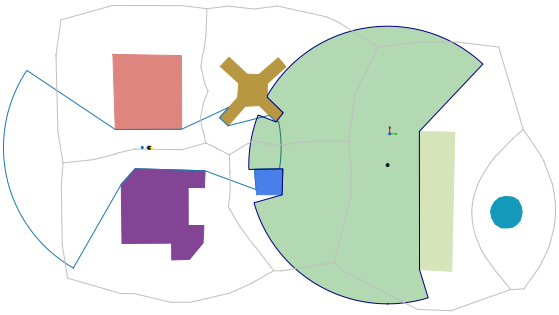
A main part of using a Network for an inspection task as in the motivating first responder scenario, is the placement of Vehicles to enable the completion of the main goal, the inspection of the building of interest. As the main inspection task involves operating an unmanned aircraft outside the LOS of the operator–and as the simulated datalink is also assumed to only function in range limited direct LOS (Section 3.3.2)–this placement hence has to establish a multi-hop communication relay between the Control Station and the Vehicle(s) doing the inspection.

The presented implementation of MVS provides a simple heuristic-based method to establish coverage to and around a building designated as the target for the inspection task. This section describes the underlying algorithms, assumptions and intrinsic shortcomings. Chapter 5 will cover a human subject study aimed at investigating the effects this aid has on the completion time of the inspection task.

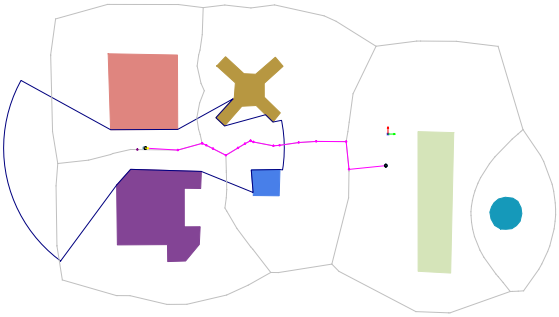
4.3.2.1 Ingress

The task of providing a functional RF coverage around a target building is split into two steps: getting to the building and then covering the surroundings of it. This section explains the ingress portion, i.e. how to establish a communication link from the Control Station to the target building.

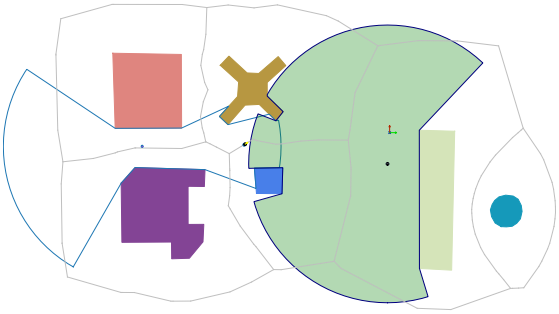
As the operational environment can be represented as a single non-simple polygon (Figure 10(d)), the sought-after RF link between the Control Station and the target building can be represented as a polyline through the interior of that polygon. Each vertex of the line would represent one blip, each line segment one RF link or a hop; the sought-after ingress polyline will be nearly identical with the com graph path connecting the Control Station



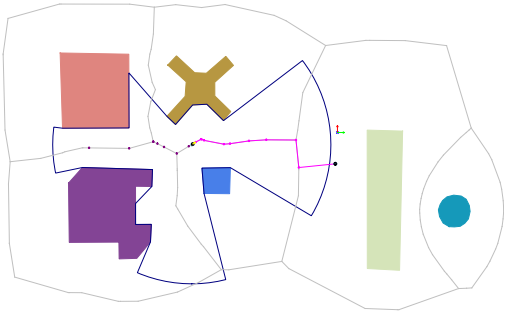
(a) A Control Station's map view after a blip just started its RTLS process.



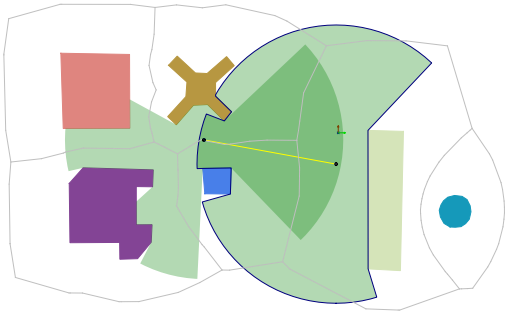
(b) A blip's map view showing the RTLS flight plan to the last known location of its Control Station.



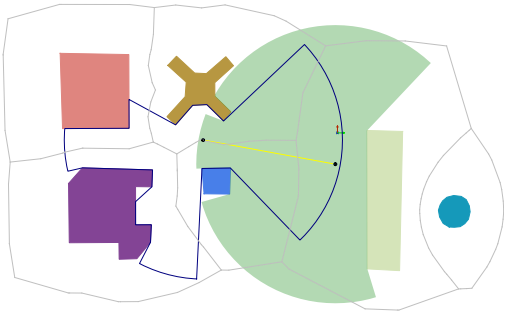
(c) The Control Station's map view shortly before the blip enters the coverage area connected to the Control Station.



(d) The Control Station's map view shortly before it can reestablish communication with the Control Station.



(e) The RTLS procedure stops when the blip is reconnected and again part of the Network's coverage area.



(f) The blip's map view visualizes its position close to the maximal range of the data link.

Figure 40: The RTLS procedure as it is visible in the map views of a Control Station (left column) and the returning blip (right column).

and the inspecting Vehicle in the Network, with slight differences at the end of the inspecting Vehicle as that will be under direct joystick control. Besides some obvious constraints on the length of the line segments (the maximum range of the RF link) and the number of vertices (up to the number of blips in the Network), another challenge is to define or find the terminating points of the line. One point is found easily, it is the Control Station's location. The other terminating point is not as obvious, as unlike with the Control Station whose position is readily simplified to a point in the environment, the target building cannot be easily represented by a single (x, y) coordinate tupel as it itself is a polygon.

Remembering how obstacles are represented in MVS, the obstacle graph (Figure 12) comes to mind as there each obstacle indeed is represented by a vertex which has been assigned a (x, y)-position in the environment: the centroid of the obstacle polygon. However, being at the centroid, this position has the drawback that it most likely is inside the obstacle and as such not actually useful. This is not only true for practical purposes (a Vehicle could not get to the location), but also for computational and numerical reasons as the location is most likely inside the hole in the environmental polygon representing the target building and as such would somewhat conflict with the underlying geometrically correct algorithms processing the environment. As no single candidate point seems immediately obvious as representing a target building, it is necessary find other candidate locations.

The ingress portion of the coverage task is to connect the Control Station with a yet to be determined end at the building site that would provide the coverage to the inspecting vehicle. As such it would be beneficial for the polyline end point at the building site if it could already provide a good portion of the coverage eventually anyways needed. Revisiting Figure 12(f), good candidates for such a location are the route graph bifurcation points around a target building. By construction these points have several benefits:

They are on the route graph. As the bifurcation points are part of the route graph, they are necessarily on it. This is beneficial as Vehicles when moving autonomously are required to move on the route graph to compensate for map errors. Moving to a

bifurcation point as such poses no need for any additional code than what is already present in the framework.

They are at intersections. If a blip is located on an edge on the route graph between two bifurcation points, the blip's FOV is essentially linear: "up the route graph" and "down the route graph", i.e. the RF coverage is limited to a single face of the obstacle polygon. At an intersection the RF coverage has at least three main directions: along all the route graph edges connected to the bifurcation vertex. A blip placed there could as such have a chance to cover two faces of the target obstacle as well as a potential ingress direction.

They are placed in open areas. By construction the bifurcation points are located at the center of the largest circle touching the surrounding buildings. This leads to the fact that all possibly LOS obstructing elements are, averaging, as far away as possible which in turn is a good heuristic for maximizing the RF covered area.

They are easily associable with obstacles. The bifurcation points are easily mappable to an obstacle via the related contextual information. Indeed, the construction process of the route graph already creates this association as Figure 14(e) highlights.

The bifurcation points surrounding the target cell are as such seen as good candidates for possible other ends to the to-be-found polyline representing the ingress communication link. With the start and end points of the line determined, what remains is the routing of the interim edges, i.e. the placement of the vertices eventually representing relaying blips. This entails finding a chain of edges that are shorter than the maximum range of the ulitzed RF link and which lie completely in the interior of the environmental polygon, i.e. which have a direct line of sight in between them.

To reduce the possible numerical complexity involved in probing or processing the environmental polygon, the placement of the start and end points of the link can be used to reduce said complexity. Figure 41(a) shows the method currently implemented: using

(a) Bifurcation points around the target and the Control Station position.

(b) Superimposing a stencil to reduce the relevant environment based on the maximum RF range.

(c) The reduced environment slot utilized for communication path finding. (253 vertices)

Figure 41: Based upon the location of the Control Station and the target, the overall environment polygon is reduced to a slot relevant for the ingress portion of the RF coverage to the target. Reducing the environment complexity can speed up geometric computations.

the start and end points of the link, a slot of the width of the maximal RF range is punched out of the environment. The underlying idea is that the closer the link polyline is to the straight line connecting the Control Station and the target (in this case indeed represented by its centroid), the better. This straight line is then simply given a "stroke thickness" of the RF range, which results in the stencil shape shown in Figure 41(b). Similarly as the operational boundary is used as an outer delimiter on creating the overall environmental polygon, the outline of this stencil is used to create a sub environment which can be used to look for the communication link. As mentioned, the motivation is that a reduced complexity of the environment utilized for the search for the communication path can speed up the computation and by the same token increase the scalability with respect to the size and complexity of the overall environment.[29]

Figure 42 highlights some alternative sub-environments that can be enabled in code. Although the round-capped slot has about 60 % less vertices than the complete environment polygon, the interpolation of the half circle ends adds vertices that might or might not be

[29] A larger overall environment has no direct effect on the searching of a communication path other than what relates to the computation of the sub-environment.

(a) A square-capped stencil trades the complexity of the circular ends for the potential inclusion of more obstacles. (232 vertices)

(b) A flat-capped stencil avoids the vertices of the half circles and additional obstacles, but can limit the target approach. (176 vertices)

(c) A stencil based upon the cell neighbors of the shortest hop connection(s) in the obstacle graph. (475 vertices)

Figure 42: Alternatives to the round-capped stencil allow for small variations in the trade off of environment complexity (roughly represented by the number of vertices of the polygon) and the covered area and (route graph) paths. (For comparison, the full environmental polygon, Figure 10(d), has 672 vertices, the round-capped stencil in Figure 41(c)) creates one with 253 vertices.)

needed.[30] Figures 42(a) and 42(b) directly approach the vertex count resulting from the half circular ends. The trade-off between circle vertices, vertices added through extra obstacles in the corners, and the stark reduction in covered area for the flat-capped case seem to be affected by the "density" of the environment or how "cluttered" it is; the presented implementation of MVS uses the round-capped slot without having done a comprehensive study.

Figure 42(c) showcases an approach to the creation of a sub-environment in a different manner, borrowing from the ideas already presented in the context of dynamic environments (Section 3.4.4). As the Vehicles creating the RF coverage will be primarily using the route graph to move (autonomous motion is bound to use the graph, only direct joystick control can traverse the environment without restrictions), the sub-environment is created not from an external geometric shape, but based upon obstacle cells as their perimeters form the route graph. The presented cell-based stencil finds the not necessarily unique shortest

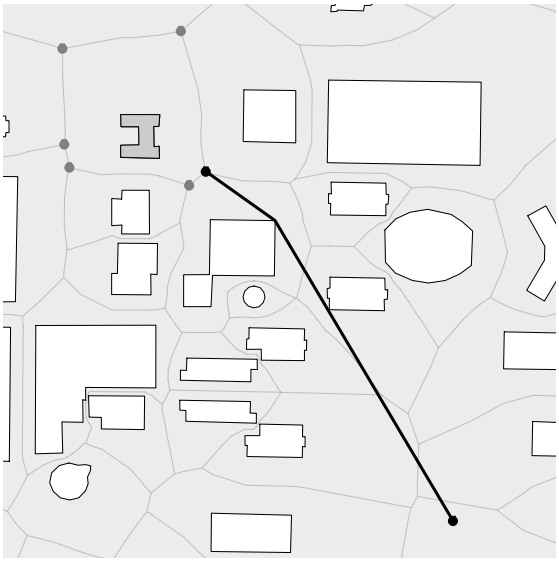
[30]The number of vertices used to interpolate the half-circular endcaps can be altered in the sources and is currently set to 20.

paths through the obstacle graph, connecting the target obstacle cell and the cell containing the Control Station, and pads them with their one-hop neighbors (compare Figure 24). The cell-based stencil hence does not reduce the vertex count as drastically as the round-, square-, or flat-capped straight-line-based ones due to its rather complex outline, but it ensures the highest coverage of actually traversable (route graph) paths.[31] However, computing this stencil is considerably more complex and might only be worth the effort if the related results can be used by other processes or algorithms, mitigating the cost of computing it.

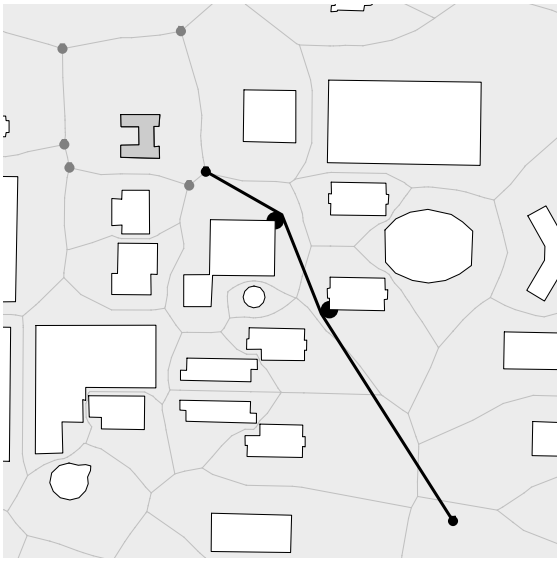
The next step in finding the ingress portion of the sought-after communication connection between the Control Station and the target is to find a polyline through the interior of the sub-environment that connects the Control Station and one of the bifurcation points. A classic approach to this would be a shortest path algorithm, especially as the environment is known. If no direct connection between the points is possible, a "shortest path" polyline will have vertices at the corners of obstacles and can be found via a visibility graph of all vertices of the environment and the communication end nodes. Figure 43(a) shows such a shortest path solution.

Given the scenario that provides the background to how the environment got to be known, however, the accuracy of the two-dimensional data with respect to the real world environment has to be taken into consideration. This would require that the obstacle polygons need some additional padding to compensate for potential location inaccuracies or motion of the Vehicles. Figure 43(b) shows how padding can affect the shortest path solution and consequently the placement and number of required interim vertices. Increasing the corner padding to allow for the highest achievable safety margin leads by construction to vertex location on the route graph as the underlying Voronoi graph exactly provides these maximum clearance routes. Figures 43(c) and 43(d) show how increasing the safety

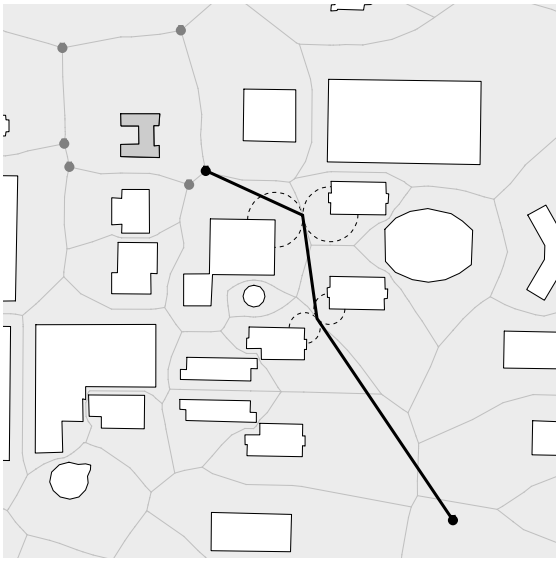
[31] Another benefit of the cell-based stencil is its preservation of the cyclic properties of the covered route graph segments; the other stencils can cause dead-end branches.



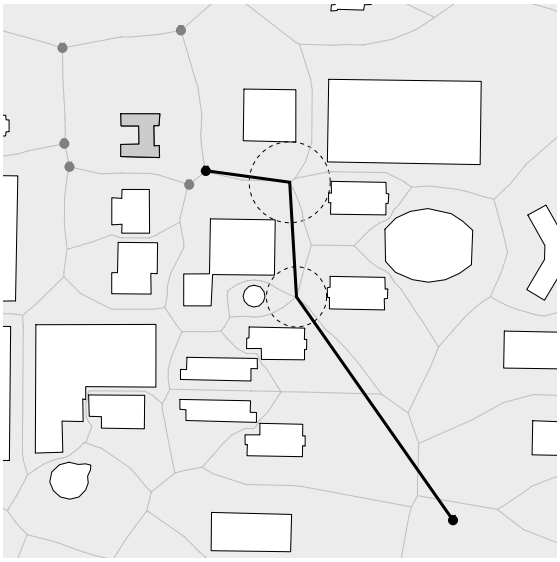
(a) The shortest and most direct path between the Control Station and one of the target's bifurcation points.



(b) Corner padding, introduced to allow for map inaccuracies and Vehicle safety margins, slightly alters the shortest path.

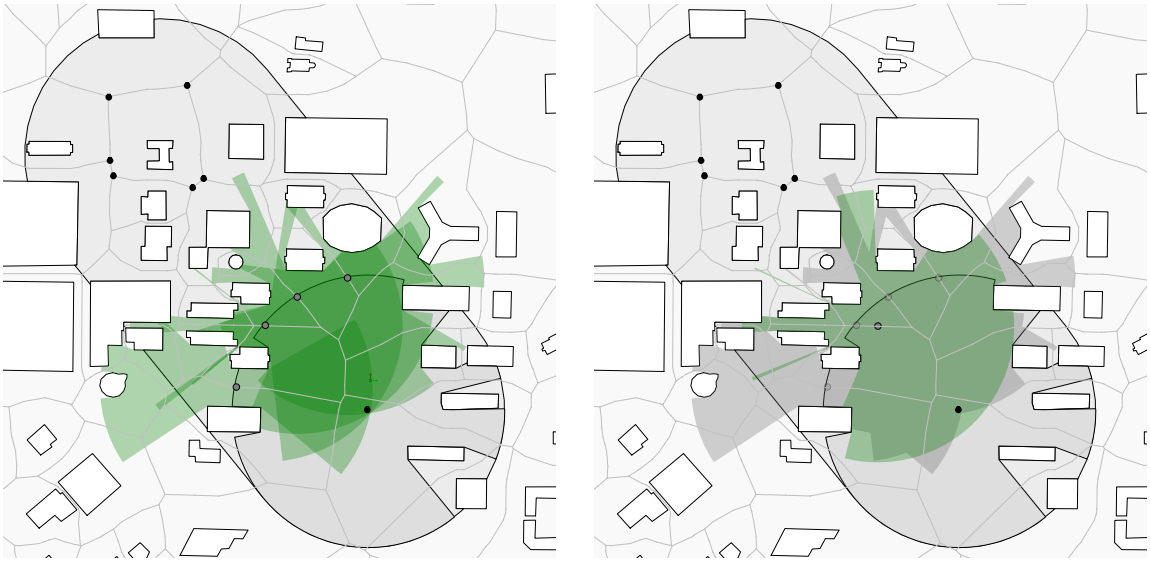


(c) Increasing the corner padding to the maximal (symmetric) distance moves the polyline vertices onto the route graph.



(d) Further maximization of the safety margin moves the polyline vertices onto the route graph's bifurcation points.

Figure 43: Starting from a shortest path between the Control Station and the target bifurcation points, safety margins for map inaccuracies or position errors stipulate moving the relaying blips at least onto the route graph, preferably even to bifurcation points as they are at the center of the locally largest inscribed circles.



(a) RF coverage at the extreme positions, i.e. at the intersections of the route graph and an RF polygon.

(b) RF coverage at an internal point, expanding what is achievable from the extreme positions only.

Figure 44: Placing relaying blips at the extremes of RF coverage will not necessarily guarantee achieving the "best" coverage as it only maximizes distance in certain directions.

padding from the obstacle corners eventually ends up at a bifurcation point after first reaching the route graph. Resulting from this observation, interim vertex location are hence also bound to lie on the route graph as that provides the safest shortest path options for Vehicle placement.[32]

Finding a polyline through the interior of the sub-environment where corner vertices only lie on the route graph is the next step. There are several approaches to this as it is closely related to classic motion planning problems. However, what sets this task slightly apart is that it is not only a pure traversal problem (although the Vehicles clearly have to get to their positions), but also a link maintenance problem stemming from the range-limited LOS datalink. One method would be the computation of a hop-distance map from the Control Station to the target points and then simply putting relay vertices at the relevant spots. Building this map, essentially propagating a wave through the environment and

[32]Note, however, that the safety padding only increases the safety margin related to uncertainties with respect to the position of the (relaying) blips; it does not necessarily increase the robustness of the communication link.

marking the wavefront every time it has traveled a distance comparable to the maximal RF range, can be computationally expensive. Unfortunately there seems to be no easy "cheat" sidestepping the continuous wave propagation by only looking at the allowed positions on the route graph as Figure 44 showcases.

Figure 44(a) shows a potential Network setup in which possible blip positions are marked at the maximum RF distance from the Control Station on the route graph. A continuous propagation would have computed the RF covered area for every possible position of a blip from the Control Station to the shown extreme locations and every area outside the the RF polygon of the Control Station (the outlined RF polygon) that is or at one time had been covered (the green shaded area) would hence be at a 1-hop distance from the Control Station. Arguing that the covered distance can only be shorter for all possible positions on the route graph that are closer to the Control Station than the shown extremes, the continuous propagation could be skipped and the coverage could be computed only for exactly those extremes. This would reduce the computational effort from the continuous evaluation of RF coverage to the computation of only a few relatively "cheap" RF polygons. In the next iterations the intersections of the route graph and the current coverage frontier would be found, the RF polygons would be computed, the covered area marked as a 2-hop distance, and the process would continue.

Unfortunately skipping the continuous evaluation of coverage to build a map will lead to a wrong map and consequently to results that are not as good as the originally replicated method might have suggested. Figure 44(b) highlights how the "cheat" skipped possibly important setups by ignoring the fact that not everything that is RF covered from a non-extreme point on the route graph is guaranteed to be covered from the extreme ends. Not only does the intermediate location push the frontier of the covered area further out, it also covers an additional bifurcation point as well as the route graph up to the frontier, allowing the next group of Vehicles to get there while being connected to the Control Station.

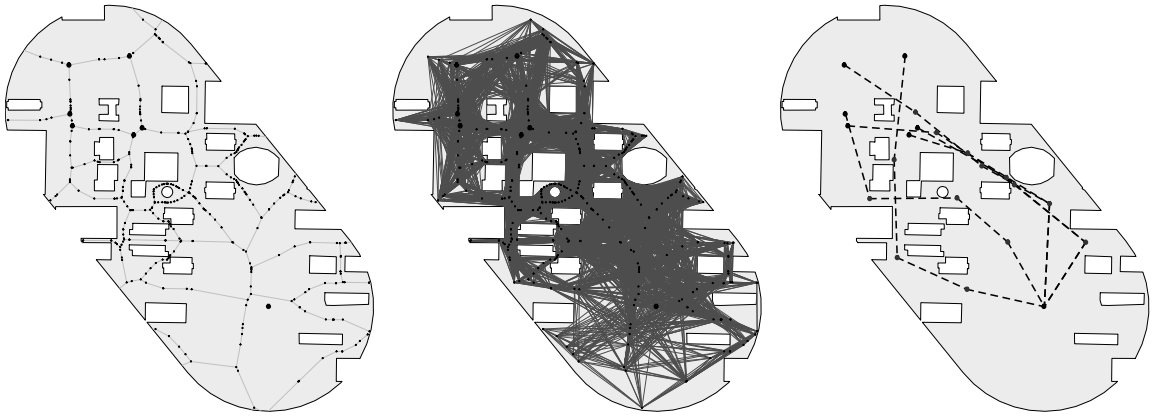
As the example shown in Figure 44 outlines, the choice of where to place the next

relay vertex isn't necessarily always at the intersection of the RF polygon and the route graph and a metric to measure a "good" versus a "bad" placement can get quickly get fairly complex as possible ways to capture the quality of a location could include, for example, the robustness of the link to position errors or the time it would take a vehicle to get to the specified location.[33] One alternative approach to the apparently required continuous evaluation of the RF coverage would be to discretize the continuous expansion of the RF coverage frontier and hence limit the number of RF polygon computations. As bifurcation points are assumed to be beneficial (see the discussion on safety padding above and Figure 43) they should be always included, leaving the edges between them to be split and (re-)interpolated. There are many possible ways to achieve this, for example, through simply dividing the path segments in equidistant pieces, possibly using a curvature measure to increase the vertex density on windy paths to allow an increased resolution in areas where position relative to the surrounding geometry might have a larger effect. The current implementation doesn't do this, but simply uses the supporting vertices of the route graph, i.e. the end points of the straight line segments making up the graph. These are on the one hand already available and on the other hand also–through construction–denser on "curvy" segments of the route graph.[34]

Figure 45 shows how the identified possible relay locations (i.e. the supporting vertices of the route graph) are used to generate a proposed set of ingress paths from the Control Station to the target building. Using the proposed positions as potential guard locations

[33] If the simple Network approximation through Δ-discs is dropped, capturing the quality of a relay location could also include estimated SNR or bandwidth, opening the optimization to problems of deciding between a few long distance hops and more shorter distance ones–which in turn would then have to include added latency and overhead added through each introduced hop. In essence, finding the "optimal" placement for such a scenario can require the previously mentioned tight(er) coupling of GNC algorithms and the utilized MANET protocol(s).

[34] The Voronoi diagram for polygonal environments is made up of straight lines and circular, parabolic, hyperbolic, or elliptical curves. VRONI [52] internally represents those mathematically correct, i.e. in a parameterized form. The interfacing API between VRONI and MVS, however, drops this parametrization and replaces all curves with straight line segments. The density of the supporting vertices as such does not directly correlate with the actual curvature of the geometrically correct Voronoi diagram, but with a change in the geometries representing them.



(a) The supporting vertices of the route graph are used as possible locations.

(b) The range-limited visibility graph of the 362 vertex locations has 10 215 edges.

(c) The algorithms proposes a minimum hop-count path from the Control Station to each of the identified target bifurcation points.

Figure 45: At the core of the ingress step to the target coverage problem is the proposition of a "best" path to each of the bifurcations points of the target's obstacle cell. The only currently employed metric is hop-count, i.e. no factors for link quality or relay position safety is taken into account.

an undirected, range-limited visibility graph is computed. For a graph with N vertices, this requires $N-1$ computations of RF polygons and $\sum_{i=1}^{N-1} i = \frac{1}{2}N(N-1)$ "within" checks, i.e. whether or not a certain location (that of another vertex) is covered by the just computed RF polygon; Figure 45(b) shows the resulting graph for the example environment. Figure 45(c) shows the result of a simple Dijkstra shortest path algorithm which is used to find a not necessarily unique minimal hop path between the Control Station and the previously determined bifurcation points around the target. These ingress path proposals are the result of the ingress step and are stored so that they can be combined with results of the following loop finding process to form a solution to the problem of establishing Network coverage from the Control Station to the target building.

The complexities of finding "better" locations for the vertex positions are also reflected in the results of the ingress algorithm: as the sole metric for finding a good path is hop-count (all edges have the same weight), the algorithm simply picks the first found minimal hop path as the solution and even stops its search when no better path can be found. That this isn't "optimal" can easily be seen when comparing the proposed ingress paths,

Figure 45(c), to Figure 43(d), an introductory figure on the merits of safety padding. As all paths with the same hop-count are currently treated as being equally "good," the algorithm picks a path that places Vehicles closer to a building than it would be necessary. Figure 43(d) indicates that there is a 3-hop path from the Control Station to that bifurcation point that has a much larger safety padding than the path chosen by the algorithm for that particular point: all that is needed is an added relay in the longest leg, possibly at the internal location pointed out in Figure 44(b). Possible improvements of the hop-count metric could include adding a vertex weight based upon the safety of the location, possibly based upon the minimal distance to the closest obstacle, or a link-quality-based edge weight, possibly based upon the minimal distance of the graph edge to the next obstacle. Another factor could be a robustness factor of the hop, for example as proposed in [6].

4.3.2.2 Loop

The next step in the process is to take care of the coverage at the site of the target. The approach taken in the currently implemented algorithm is based upon the approach that if the complete area around the target has Network coverage than the inspecting unmanned aircraft would not have to care about maintaining a link with the Control Station and can move freely in the surroundings of the target.

There are many classic coverage problems in computational geometry (see, for example, [72]), all making slightly different assumptions of what is to be observed (or guarded), whether the observers (or guards[35]) are stationary or mobile, whether they have to see each other or not, *etc. pp.* Following the nomenclature of [72, 76], the problem to cover an obstacle cell would most likely be phrased as "point guards to cover the interior of a polygon with holes." Limiting the guards to the edges (i.e. the route graph), the "point guards" would become "edge guards," although this nomenclature might allow placing guards at the edges of the interior holes, i.e. the walls of the obstacles, which is something that shouldn't

[35]The notion of a *guard* results from a famous visibility problem asking how many guards are necessary to guard the interior of an art gallery.

be allowed. For the "fortress problem with guards in the plane" (i.e. guarding the outside of a polygon in an infinite plane) there exist an always sufficient and sometimes necessary upper bound of $\frac{N}{3}$ guards for a fortress (i.e. the target building) of N vertices.[Theorem 3.15, 76, p.991] This, however, does not take into account that the guards in the presented case have a limited sight distance, the maximum range of the datalink and is hence only of very limited use. As art-gallery-related problems do not take a range limitation into account, i.e. guards are conventionally assumed to be able to see till infinity, or a wall, many of the established results can hence not be used directly in the context of MVS. Additionally, the theoretical results deal with generic and arbitrary polygons of N vertices and often only provide upper or lower bounds on the number of guards, whereas the sought solution for the coverage problem of an obstacle cell requires the coordinates of the guard locations. Circumventing this issue, MVS doesn't provide a provably correct algorithm to determine the placement of guards for generic obstacle polygons, it just uses a simple heuristic that is generating functioning proposals reasonably often.[36]

The utilized heuristic is working as for the purpose of the communication coverage problem in the present case, several helpful assumptions can be made to simplify the problem of covering the area surrounding the target:

Treating the problem as one polygon in an infinite plane. The goal of the site local coverage is to cover the surroundings of a single target building. The obstacle cells– through construction– partition the overall environment space into parcels associated with an obstacle, making it the sole polygon in its area. As the obstacle cell boundaries, again by construction, are at the halfway mark between obstacles, it can be assumed without loss of generality that the inspecting Vehicle will be operating inside the obstacle cell associated with the target building. As such, coverage outside the obstacle cell is irrelevant as long as the complete inside is covered. The outside is

[36]The claim of functioning "reasonably often [enough]" is not based upon a methodological approach, but simply through experiments with the trial maps used for the human subject study presented in Chapter 5.

hence ignored and any obstacles other than the target are not specifically considered. This effectively renders the problem as a single polygon (target) in an infinite plane (obstacle cell and the area outside of it) which is known as the "fortress problem with guards in the plane."

Assuming all obstacles as being convex. Although this is clearly not the case (for example, the building used as a target so far is not convex), most concavities of buildings seem to be negligible for the inspection task as it most often is sufficient for the inspecting Vehicle to follow a path along the convex hull of a building and still obtain enough data.[37] As the human operator can always alter the positions of Vehicles, the operator can correct cases where this assumption does not hold and manually correct the Network if a certain building cavity cannot be sufficiently explored.

Limiting guard locations to the route graph. The treatment of the problem as a "fortress problem" stipulates the use of "point guards" as the only remaining edges would be the outside of the target polygon. This requires, however, the use of geometric approaches to find guard locations as there are infinitely many possible point locations in an infinite plane. The area can be discretized, for example through a simple grid, but the same safety arguments made in the ingress section hold here as well, which stipulates limiting the possible guard locations to the route graph.

The fortress problem with guards in the plane can be drastically simplified if the fortress is a convex polygon in an infinite plane: assuming that guards can see infinitely far, the exterior of any convex polygon can be covered by two guards if they can be placed far enough away from the polygon; if the guards have to see each other, three is always sufficient by spanning a large enough triangle around the polygon and placing the guards at the tips. This

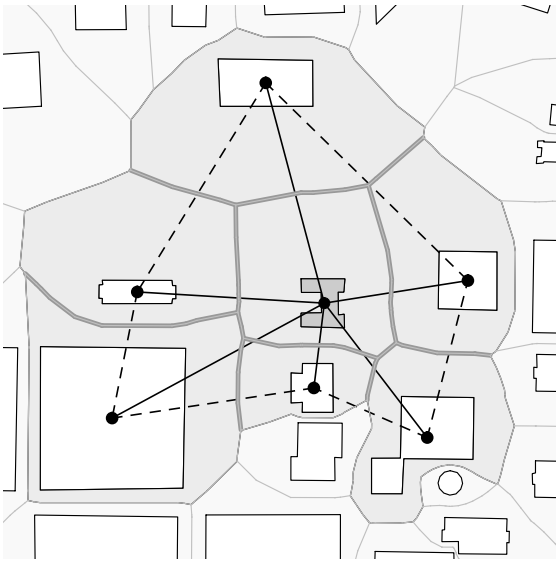
[37] The question of whether or not a building concavity needs to be entered could be answered through looking at the resolution of the utilized scanner. The underlying thought is that the utilized sensor, for example, a simulated camera, has a certain angular resolution, stemming from its FOV and image resolution. In combination with a requirement for smallest resolvable detail this can be used to compute a maximal allowable distance.

clearly isn't always possible in the given coverage scenario of the obstacle cells as there is a range limitation and obstacles aren't necessarily convex, but the underlying principle of the three guard case still applies: placing guards at the vertices of any closed polyline surrounding the (convex) polygon will cover the exterior of that polygon. Using the assumptions made above, the problem of finding the positions of the "guards in the plane" can hence be reduced to finding a closed polyline that has its vertices on the route graph, whose edges are not longer than the maximum RF range, and that encloses the target building.

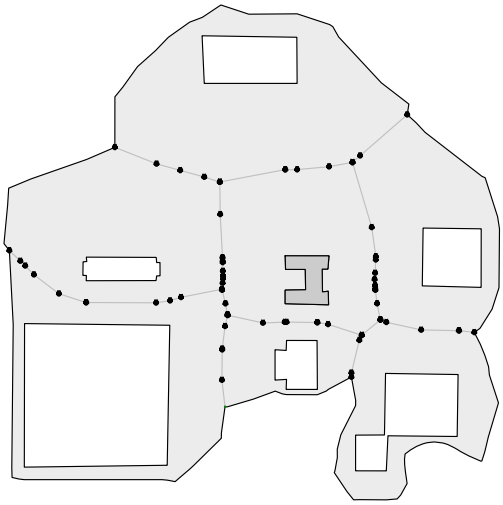
Figure 46 highlights the approach taken by MVS to solve the problem. The first step is to determine which segments of the route graph should be allowed beside the obvious outer perimeter of the relevant obstacle cell. As bifurcation points, as previously mentioned, provide a good location for RF coverage as they are the centers of the locally largest inscribed circles, adding the segment that branches there from the obstacle cell perimeter provides more most probably favorable choices. Following the same argumentation as for creating the sub-environment for the ingress portion that stated how a cell-based stencil maximises the covered route graph segments, Figure 46(a) shows how the related construction of a target (sub-)environment using the 1-hop neighbors of the target can simultaneously identify all relevant sections of the route graph (the segments dual to the obstacle graph edges) and create a target environment which covers all those segments. (Compare the cell-based ingress stencil shown in Figure 42(c).) Also following the same argumentation as in the ingress portion, the vertices supporting the route graph are selected as potential guard locations, Figure 46(b).

The next step would then be to compute a RF graph to find the loops around the target. Finding those loops, however, needs to be prepared in order to allow for an efficient way of finding them. There exist algorithms to find loops in very generic graphs and identify and list them,[38] but the loop portion of the overall process is only concerned about loops

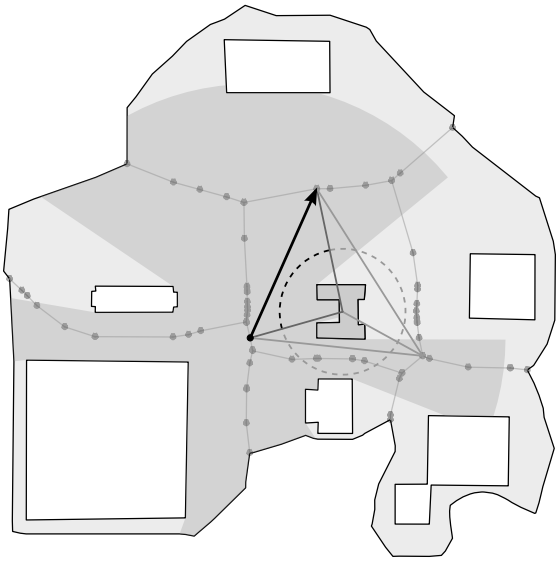
[38]The graph specific library used by MVS, Boost.Graph [81], provides for example Hawick's algorithm.[51]



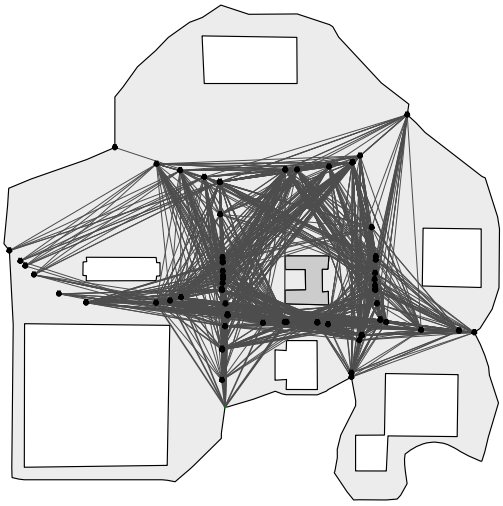
(a) The target obstacle cell and the 1-hop neighbors form the target environment. The route graph segments dual to the relevant obstacle graph edges give the permissible guard locations.



(b) The vertices supporting the Voronoi-based route graph are used as discretized locations along the permissible route graph segments.



(c) Using the centroid of the target as the pivot point, the (clockwise) directed range-limited visibility edges can be computed. The inner angle covered is added as an edge weight.



(d) The 77 vertices result in a graph with 1653 directed edges. A depth-first search can be used to find loops around the target by comparing start and end when the accumulated edge weight has reached 360°.

Figure 46: Using the supporting verticies of the route graph a directed range-limited visibility graph is created in the target environment. In the process of determining the direction of a link, the covered inner angle with respect to the centroid of the target is computed in order to find loops in the graph which encircle the target building once.

with a winding number of ±1 around a specific point in the environment, the centroid of the target building. As such, MVS tailors the RF graph generation in order to avoid the potential overhead of using a generic loop finding algorithm. Figure 46(c) depicts how the "within" check performed to establish whether or not an edge between two vertices exists is expanded through determining the direction of an edge with respect to the centroid of the target. For the directed RF graph computed here, only one direction would be allowed (clockwise in the example shown) and the cross product computation used to determine the direction is also used to get the inner angle around the centroid covered by the edge; the found angle is set as the weight of that edge.

With the prepared directional RF graph, finding single loops around the target is straight forward. A depth-first search starting at any vertex just needs to accumulate the edge weights of the traversed edges and compare the vertex reached at 360° to the start vertex: if the vertices are identical, then a loop is found; otherwise the depth-first search can be terminated as no loop with a winding number of ±1 around the target will be found.[39] Unfortunately, as Figure 46(d) might convey, there is a potential for a large number of possible loops, even with the minimal number of only three supporting vertices. As MVS in the presented form has no other metric for a found loop than the number of hops, this bears the potential to skip computations as all 3-hop loops are deemed equally good. Combining this with the need to match the loop paths with the ingress paths, the last step in the loop portion is to use the above described algorithm to find a (not necessarily unique) minimal hop loop originating at every bifurcation point of the covered route graph segments. These loop path proposals are the result of the loop step and are stored so that they can be used with the ingress step proposals to form a final proposed RF formation to cover the area surrounding the target building.

Computing the directed RF graph and finding the loops around the target currently are

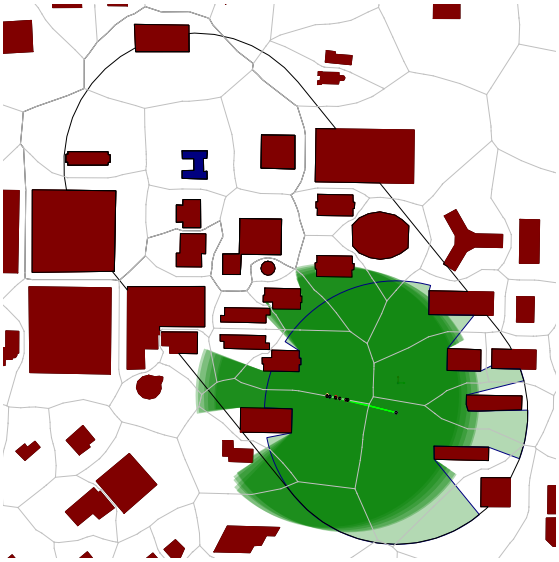
[39]There might be a computational advantage to sort the outbound edges by descending weight. That way a depth-first search could find smaller hop loops quicker as those on average have higher weights on their edges.

the computationally most expensive element in MVS. Creating the loop proposals takes a considerable amount of time (in the order of seconds on a current desktop computer) which includes, unfortunately, some repeated work as loops consisting of N vertices will be found (and computed) up to N times: once for each search starting at one of the vertices. Reducing the number of "start vertices" to only bifurcation points eliminates this problem to a certain degree as as the same loop will now only be found for each supporting bifurcation point. Another cause of duplicated work stems from the overlapping of the ingress and target environments. As the same source pool of vertices is used (the supporting vertices of the route graph), edges and the related "withins" are computed twice, if once though directed and once undirected. Improving this is important to increase the future scalability of MVS as even for relatively small teams of eight to ten vehicles (as used, for example, during the human subject study presented in Chapter 5) the need to use several hosting computers might arise.

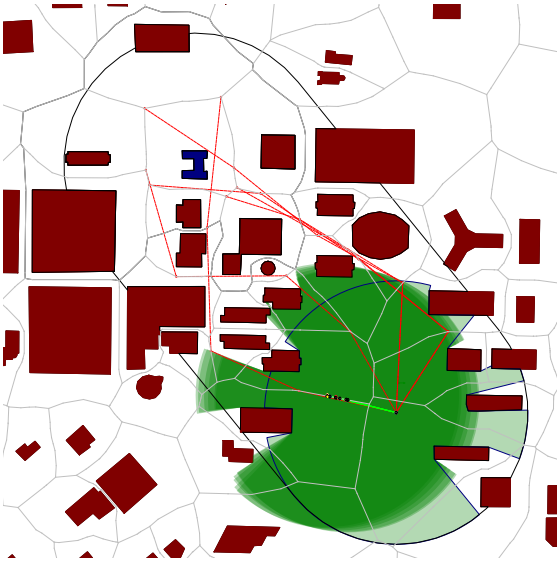
4.3.2.3 RF Formation Proposal

The last step in providing the operator with an aid to support the creation of Network coverage around the target obstacle is to combine ingress and loop path proposals and determine the "best" among them. In the presented implementation of MVS the ingress as well as the loop path proposals all can be associated to one of the bifurcation points around the target. Ingress and loop paths sharing the same point are simply combined there and all the resulting paths are ordered by total hop count. The first path with the minimal number of hops is then deemed the best solution and is presented to the operator.

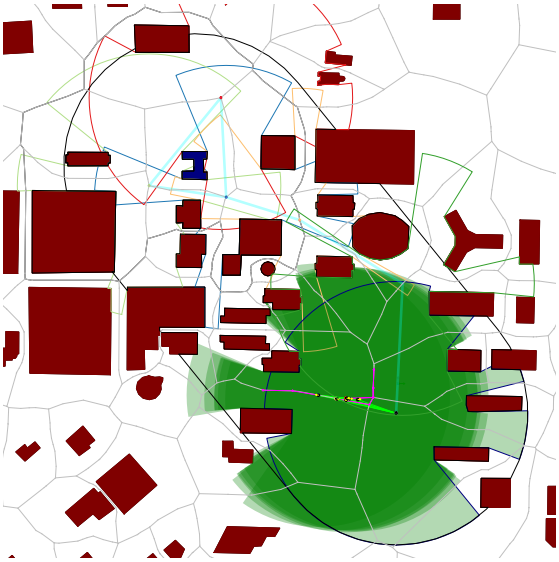
Figure 47 depicts how the use of the obstacle coverage aid is shown in the map view of the operator's Control Station. The operator starts by marking a building as the target through the buildings context menu. This changes the color scheme of the map view to single out the target building and computes and shows the round-capped ingress environment as well as the target environment, Figure 47(a). The next step by the operator is to



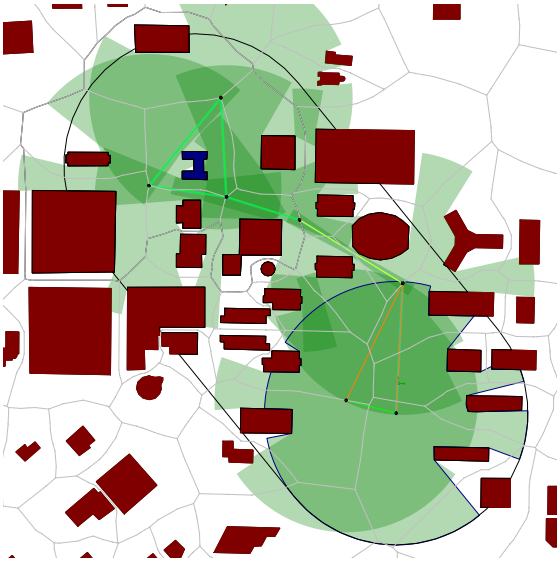
(a) The initial situation after a target has been identified: the obstacle color scheme singles out the target and the ingress and target environments are shown as visual aids.



(b) As the loop computation can take a while, the quickly available ingrass path proposals are shown to give the operator an initial idea on which paths the heuristic considers.



(c) The cyan line depicts the final formation proposal after the ingress and loops paths have been combined. When the operator realizes this formation, MVS sends the closest Vehicles to the identified locations.



(d) The final Networkcoverage is relatively complete, even in the target's cavities. With the shown formation the operator can pilot a Vehicle from the Control Station to the target and inspect the complete building envelope.

Figure 47: The proposed formation of the obstacle coverage aid creates an area coverage around the target building that is connected to the Control Station.

start the formation proposal computation through the context menu of the target building. This triggers the computation of the ingress and loop paths as well well as the final formation proposal. As the computation of the loop segments can take a while, MVS shows the interim ingress solutions while no better results are available, Figure 47(b). Experienced operators can take this as a hint to which routes are potentially picked by the heuristic and could start deploying vehicles in that direction. When the loop paths are available, the selected "best" combination with an ingress path is shown as the final RF proposal formation, the cyan lines in Figure 47(c). The operator can chose to completely ignore this, or, again through the target building's context menu, trigger the realization of the proposed formation. MVS will then process the formation relay positions starting at the Control Station and select the blip with the shortest route graph distance to that location from all blips that are autonomous (compare Figure 25). Figure 47(b) depicts the proposed formation shortly after the realization has been commanded and Figure 47(d) shows the Network coverage after all Vehicles have reached their commanded positions.

4.4 Discussion

In conjunction with the presented aids, the lost link RTLS procedure and the coverage aid, the presented GUI utilizes and expands the MVS framework to enable a single operator to inspect a target not only beyond LOS, but also beyond single hop RF range. The interfaces available to the operator allow a higher level control of Vehicles through a simple drag and drop mechanism as well as direct joystick control for a more precise positioning whilst also enabling the operator to plan Vehicle positioning based upon live RF coverage predictions.

The GUI and the coverage aid aim at increasing the operator's SA through an increased predictability: while limiting the autonomous motion of Vehicles to the route graph reduces placement options and increases travel time, it increases the operators understanding of a Vehicles path by simply looking at the Vehicle's current location and its target and "connecting the dots." In combination with the RF coverage prediction polygons, allowing a

lost link state for a limited time, and the RTLS procedure to recover from longer lost link situations, this predictability allows the operator to plan and improve Vehicle placement for an increased and optimized Network RF coverage.

SA related to the state of the stability or robustness of the Network communications is supported through two graphical elements in the map view: the com graph indicates point-to-point connections in the Network and uses a color coding of its edges to warn an operator of links at the range limits. The same information is additionally encoded in the green shading of the RF covered area as RF polygon intersections are shaded darker. A blip located further away from the boundary of an intensity change in the green shading as such provides a more robust link than one closer to such a boundary. As the intensity is also correlated to the number of edges in the com graph, it also indicates that moving a blip located in such a darker area affects more communication links than moving a blip located in a lighter shaded area.

The presented GUI and the aids also highlight some intrinsic problems related to the collaboration of a human operator and autonomous unmanned aircraft. An example related to the *What is it doing?* question is the communication of the "state" of a Node in the Network. The GUI tries to capture some of the most pertinent information directly in the graphical representation in the map view, aided with some textual information in the data browser, but it seems impossible to capture the information contained in a state chart in a single line listing sates in the browser. Codifying more information in the form of graphics also is limited as an operator can only process so many combinations of shape and color changes as information carriers.

The *Why is it doing it?* question is a little more involved as the question for a causal explanation could require looking at the immediate past, checking if prior events could have caused the currently observed state. An example for this type of situation could be the RTLS process: if a Vehicle hasn't had contact to its Control Station it plans a path back to the last know location of it and stops as soon as a link with the Network is established. The

rational for this behavior comes from the assumption that if a Vehicle was sent to a location, the operator most likely did that for a particular reason. As this reason is not known to the Vehicle (the only currently conveyed information is “go there and stay”), the autonomous Vehicle simply “doesn’t know” what to do and, once having reestablished contact with its Control Station, simply loiters, waiting for new commands. As the RTLS process has been triggered, there is a chance that the operator was not aware of the underlying flaw in the original reasoning to send the Vehicle to its location. This leads to the situation where at some point the operator notices the Vehicle at a location where it wasn’t send to (it has relocated to reestablish link), but still is connected. Particularly during the deployment phase of a mission, this could trigger the question why the Vehicle is at the now observed location and not at the location it was originally send to be at–if the operator remembers if and to where that particular Vehicle had been sent earlier.

For the presented implementation of MVS the *How do I change it?* problem is not particularly pronounced as both, the drag and drop and the joystick interface, are globally available. An operator can at any time use any of the two methods to take command of a Vehicle without the need to know the current state it is in.

4.5 Possible Expansions

The presented methods assume that Vehicles can hover. The inclusion of hover incapable aircraft requires HMI changes in the joystick interface as well as an adapted loop and ingress path finding method. The loop portion of the coverage could be converted to a continuous motion cycle: the Vehicles assigned to the loop could simply circle the target continuously, either on the obstacle cell perimeter or only using the proposed locations. The bigger issue here is to maintain the connection to the Control Station as the ingress portion isn’t a cycle. If the link robustness allows it, fixed wing aircraft could loiter at the proposed locations. If this is not possible, a completely different approach to establishing the ingress has to be found.

As the GUI and the presented aids expand the underlying code framework to include the human operator, there are many HMI related elements that could be of interest. A big issue in the context of MANETs could be how to deal with the possible introduction of a lag in the remote piloting interface or how to better convey information about the state of an unmanned aircraft to the operator. Interesting possibilities could also stem from having two human operators collaborate: how could they "share" Vehicles (they currently can be transferred through a sequence of join and leave commands from different Control Stations), how would they divide their tasks (one operator could for example focus on maintaining the coverage while the other one performs the inspection), and how could they exchange mission pertinent information about the tactics and strategies they plan to employ whilst possibly not being collocated.

More software development focused follow up work could investigate the benefits of the different stencil shapes or how to incorporate more advanced metrics than the simple hop-count into the selection algorithms for the proposed loop and ingress paths. A very interesting step would also be to rework the centralized coverage aid path proposal into a distributed algorithm; the latter is certainly very interesting in the context of equipping the unmanned aircraft with more information on the reasons for why they are sent by an operator to a certain location as this information could then be used overcome the "what to do now" situation after having recovered from a lost link state.

CHAPTER V

HUMAN SUBJECT STUDY

The presented implementation of MVS provides two aids that are directly aimed at supporting a single operator managing a team of supportive unmanned aircrafts while performing a mission critical task with a primary unmanned aircraft. The human subject study presented in this chapter is aimed at testing whether or not the obstacle coverage mechanism presented in Section 4.3.2 can indeed help a single operator to reduce the completion time of a simulated mission.

5.1 Design of Experiment

The experiment is created around a setup comparable to the motivating first responder scenario: the participants are tasked to use a team of unmanned aircraft to inspect a specific building.[1] To do that, the participants, taking the role of the operator, are presented with an environment, a team of unmanned aircraft, and the building they are to inspect. The mission is timed: the clock is started when they are told which building is the target and it is stopped once the target has been completely inspected. The inspection requires the participants to "scan" all faces of the target building with a Vehicle under joystick control while not being further than 10 m away from it. The time to complete the inspection is taken as a performance measure, assuming that faster completion time represents a better performance as that would hopefully lead to a faster completion of the overall first-responder mission. To evaluate the helpfulness of the aid, the mission completion times of runs using the coverage aid are compared to those not using the aid.

A single run of the experiment is generally structured as follows:

[1] Appendix B.1 lists the wording used during the initial briefing of the participants in "3 Motivational Scenario."

1. The participant is given a simulated environment, i.e. a certain number of Vehicles distributed close to a Control Station in a map.

2. The participant is told which obstacle represents the target building to be inspected. (The experimenter starts the timing.)

3. The participant will arrange Vehicles as to enable the joystick controlled inspection of the target, i.e. to ensure Network coverage. (This is done either with or without the help of the coverage aid.)

4. The participant will inspect the target with the primary unmanned aircraft, possibly with interruptions to alter the positions of the supportive unmanned aircrafts in order to maintain Network coverage.

5. Once all faces of the building have been "scanned", the run is over. (The experimenter stops the timing.)

The experiment aims to test whether or not the availability of the obstacle coverage aid improves the performance, i.e. if the obstacle coverage aid reduces the completion time of a run. The experiment does not, however, enforce the use of an obstacle coverage approach to solve the problem of operating the primary unmanned aircraft on the far side of the building.[2]

5.1.1 Statistical Setup

The presence of the obstacle coverage aid is certainly only one of many factors which could potentially impact performance, measured via the completion time. Beside the presumably most important factor, the operator's skill to interact with MVS and the piloted unmanned aircraft, two factors that are somewhat more easily controllable are the number

[2]The participants are shown several tactics on how to enable Vehicle operation beyond LOS of the Control Station during the training phase of the experiment. Question 3.2 of the post-scenario questionnaire attempted (Appendix B.3) to capture the participants intention.

Table 1: The scenarios to be tested in the human subject study. As the aid presence it the factor of interest, it is retained in the scenario labeling. The map and UAV factors are combined into a single factor *Environment* with four levels: $\{(M,N),(M,n),(m,N),(m,n)\}$, mapped to $\{1,2,3,4\}$.

(a) Environments.

Environment	Map	UAVs
1	M	N
2	M	n
3	m	N
4	m	n

(b) Scenarios.

Scenario	Aid	Map	UAVs
A1	A	M	N
A2	A	M	n
A3	A	m	N
A4	A	m	n
a1	a	M	N
a2	a	M	n
a3	a	m	N
a4	a	m	n

of unmanned aircraft available to the operator and the "complexity" of the environment. The number of unmanned aircraft is easily defined as the number of Vehicles that are available in the simulation.[3] As the "complexity" of an environment is not as easily measured, the density of obstacles is used as a surrogate. The experiment is setup as a full factorial design with three factors, each at two levels. The three factors are:

Aid Presence. Potential levels are *A*, when the obstacle coverage aid is present, and *a*, when the aid is not present.

Map Complexity. Potential levels are *M*, for a more complex map, and *m*, for a simpler map.

Number of unmanned aircraft. Potential levels are *N*, for more unmanned aircraft, and *n*, for fewer unmanned aircraft.

[3] All Vehicles in the study were GustUav which were tied to GUST's default simulation of a Yamaha R-max helicopter.

As the effect of the presence of the coverage aid is the driving research question for the human subject study, this factor is tested *in subject*. Combining the factors *Map Complexity* and *Number of unmanned aircraft* into the single factor *Environment* (mapping $\{(M,N),(M,n),(m,N),(m,n)\}$ to $\{1,2,3,4\}$), the resulting scenarios are listed in Table 1.

To test the aid *in subject* all participants are hence given the same set of environment levels, once with the coverage aid being present and once not. To eliminate possible effects resulting from the sequence of the environments, a non-repeating Latin square setup is chosen, i.e. the order of the sequences is given by the rows of a square matrix $L_4 \in \{1,2,3,4\}^{4\times4}$ in which each pair of elements $(l_{i,j}, l_{i,j+i}), i \in [1,4], j \in [1,4)$ is unique.[4] One possible such row-complete Latin square of order four is

$$L_4 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}.$$

A resulting perfect design would hence require at least four participants, who each complete all four environments twice, once with the aid present and once without. However, running each participant through the complete set of eight scenarios was deemed to taxing on the volunteers' time and it was elected to limit the number of scenarios presented to each participant to four, i.e. two environments with the added factor of the presence of the placement aid, which is as such kept *in subject*.

As the rows of the Latin square L_4 do not contain repeating pairs, each row represents three pairings of environments, leading to the total of the $4 \times 3 = 12$ expected pairs as order matters. Including the *Aid* factor with the created environment pairs hence gives the individual experiment runs for the minimally required twelve participants. Table 2 lists how the scenarios are mapped to experiment run sequences.

[4]For some practical data on Latin squares see Appendix C.1.

Table 2: The sequence of experiment scenario runs as it can be mapped to participants. A total of at least 24 participants are required to completely cover the experiment design space. If more than 24 participants can be recruited, the assignment would overflow, i.e. the 25^{th} participant would get the sequence *S1* again.

(a) First stage sequence set (aid second).

Sequence	**Experiment Run**			
	1^{st}	2^{nd}	3^{rd}	4^{th}
S1	a1	a2	A1	A2
S2	a2	a3	A2	A3
S3	a3	a4	A3	A4
S4	a2	a4	A2	A4
S5	a4	a1	A4	A1
S6	a1	a3	A1	A3
S7	a3	a1	A3	A1
S8	a1	a4	A1	A4
S9	a4	a2	A4	A2
S10	a4	a3	A4	A3
S11	a3	a2	A3	A2
S12	a2	a1	A2	A1

(b) First stage sequence set (aid first).

Sequence	**Experiment Run**			
	1^{st}	2^{nd}	3^{rd}	4^{th}
S13	A1	A2	a1	a2
S14	A2	A3	a2	a3
S15	A3	A4	a3	a4
S16	A2	A4	a2	a4
S17	A4	A1	a4	a1
S18	A1	A3	a1	a3
S19	A3	A1	a3	a1
S20	A1	A4	a1	a4
S21	A4	A2	a4	a2
S22	A4	A3	a4	a3
S23	A3	A2	a3	a2
S24	A2	A1	a2	a1

The choice of starting the sequences without the placement aid is deliberate as the presence of the aid could guide the participants into a certain way of approaching the scenario. Not exposing the participants to the aid's solution is hence deemed more important than any potential overlap of performance increase through the presence of the aid and the additional training the participants get throughout the course of the experiment sequence. As a consequence, all participants have to be trained to proficiency in interacting with the simulator before they engage in the actual experiment runs as only then it can be assumed that repeating a set of environments does not significantly improve the participants' performance.

If training the participants to proficiency is not achieved, a confounding exists between a potential training effect through the already finished experiments and the presence of the aid in the later experiments. To counteract that, the order of the experiments would have to be permuted, which would drastically increase the number of experiment sequences and hence the number of required participants. When the order does not matter, there are

$\binom{4}{2} = 6$ possible pairs of environments, $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$. As before, including the presence and non-presence of the aid, a total of four scenarios result from each pair of environments. For example, the environment pair $(1,2)$ has the four related scenarios$\{A1, a1, A2, a2\}$. Checking for all possible orders amongst those six sets of four elements would hence result in the need for $6 \times 4! = 144$ participants if each participant is only to do a sequence of four experiments. This is obviously rather infeasible. As an alternative to this fully randomized setup, a second set of runs is crated by duplicating the order of the *Environment* factor from the first set (Table 2(a)), but now the aid is present initially and then disabled in the later runs (Table 2(b)).

The creation of the second sequence set allows for a staggered setup of the experiment. After the first twelve participants have completed the experiment a decision can be made whether the training to proficiency was indeed achieved or whether a confounding of the results is probable. Depending on that, the next participants can be either given the first stage sequence set again for a repetition, or the second stage sequence set to correct for the confounding. As a result, a total of at least twelve, but possibly 24 participants are needed to complete the experiment.[5]

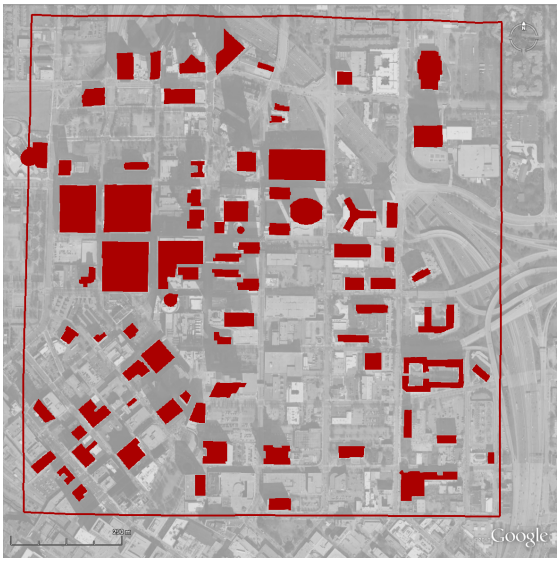
5.1.2 Experiment Scenarios

Figure 48 depicts the environments selected as the two levels of the *Map Complexity* factor. The target building is not tied to the environment, but to the combination of the environment and the number of available unmanned aircraft. This setup eliminates the case where participants could try to redo an earlier setup in cases where their sequence would include environments which only differ in the number of unmanned aircraft. For example, sequence *S1* uses the environments *1* and *2*, which both use the "complex" map, but differ in the number of available unmanned aircraft (compare Table 2 and Table 1).

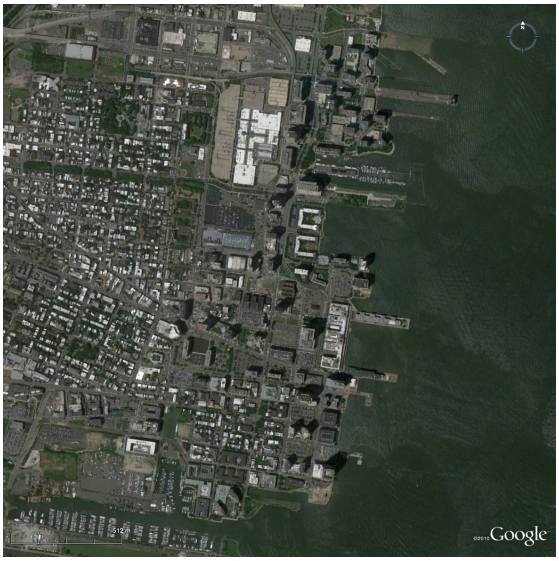
The levels chosen for the *Number of unmanned aircraft* factor are four for the lower

[5]Training to proficiency was not achieved during the experiment; a total of 24 participants were needed to complete both sequence sets, see Section 5.3.

(a) The more complex map is based upon a downtown segment of Atlanta, GA, USA.



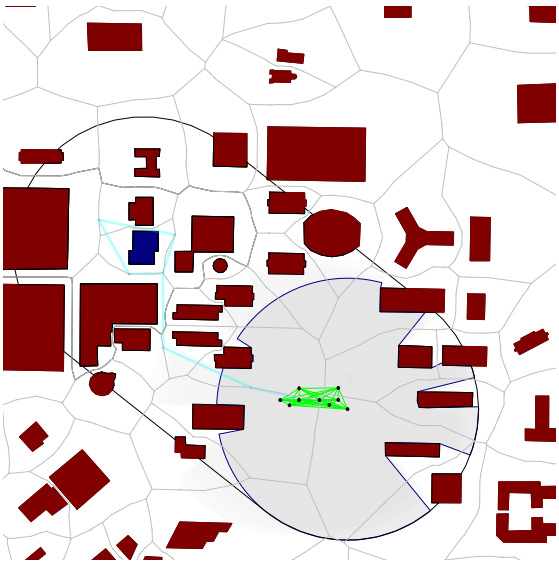
(b) The resulting Atlanta environment has 76 buildings in a roughly 1250 m × 1300 m area.



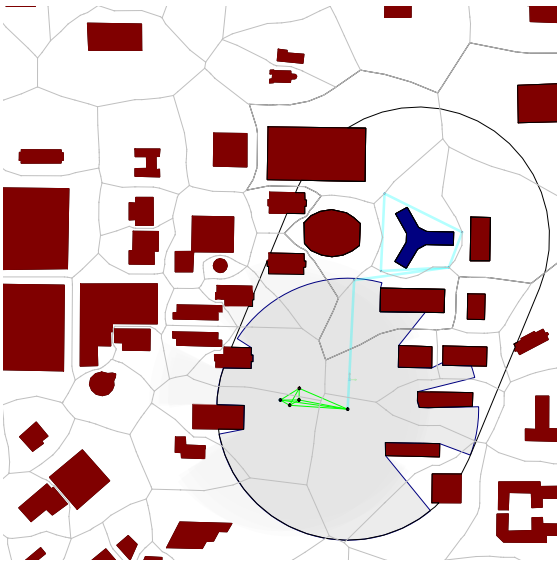
(c) The less complex map is based upon a shoreline segment of Jersey City, NJ, USA.

(d) The resulting Jersey City environment has 58 buildings in a roughly 2800 m × 1150 m area.

Figure 48: The two different scenario environment maps used during the human subject study. The Atlanta environment (top row) is used as the "complex" map as it is smaller and denser than the Jersey City environment (bottom row), which is used as the "simple" map. Satellite Imagery © 2010 Google.

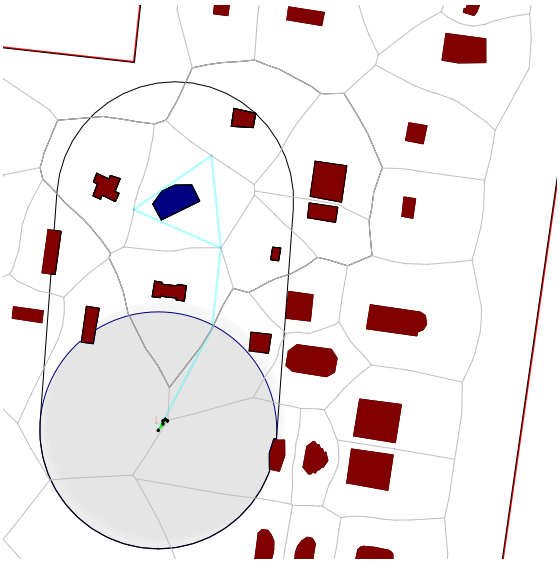


(a) Environment *1* (M, N): complex map (Atlanta), more unmanned aircraft (8). The coverage solution proposed by the aid requires 6 Vehicles.



(b) Environment *2* (M, n): complex map (Atlanta), less unmanned aircraft (4). The coverage solution proposed by the aid requires 5 Vehicles.

(c) Environment *3* (m, N): simple map (Jersey City), more unmanned aircraft (8). The coverage solution proposed by the aid requires 6 Vehicles.



(d) Environment *4* (m, n): simple map (Jersey City), less unmanned aircraft (4). The coverage solution proposed by the aid requires 4 Vehicles.

Figure 49: All environments use a different target building (blue) in order to avoid any training effects. The scenarios with more unmanned aircraft can be "solved" by following the solution proposed by the obstacle coverage aid. The scenarios with fewer unmanned aircraft (n) require a manual intervention as the proposed solution is not realizable with the given number of Vehicles (brittle automation). However, all scenarios can be "solved" through a complete coverage approach, see Appendix C.2.

level ($n \triangleq 4$ Vehicles) and eight for the higher one ($N \triangleq 8$ Vehicles). These numbers were chosen in combination with the respective target buildings (see Figure 49) so that a variation in "difficulty" was achieved where, based on the opinion of the designer, environment 3 is the easiest, followed in increasing complexity by 1, 2 and 4. Appendix C.2 elaborates on the perceived individual complexities of the scenarios as well as on possible solutions.

5.2 Experiment Execution and Data Collection

The experiments were conducted with each participant individually in an office cubicle environment at the Georgia Institute of Technology. All experiments were conducted within a three month period at a time chosen by the participants. The experiment station consisted of a desktop computer (Lenovo Thinkcenter M, Intel Core i7-3770, 4GB RAM, openSUSE 13.1) with two 24 inch screens, two RealFlight Interlink Elite controller (one in *Mode 1*, one in *Mode 2*, Figure 39), mouse and keyboard. In order to reduce the computational load to enable screen recording of the individual experiments, an additional portable computer (Dell Alienware M17x R3, Intel Core i7-2670QM, 8GB RAM, openSUSE 13.1), connected via gigabit ethernet, was used to run some of the Nodes.

Each individual experiment followed the same script (Appendix B.1) in order maximize uniformity across all participants. After an informed consent was obtained from the participants (Appendix B.2), the motivating scenario was read to the participants (Appendix B.1, "3 Motivational Scenario."). After that, participants' preference for a controller setup was ascertained (i.e. whether to use *Mode 1* or *Mode 2*) and they were trained on how to use MVS in a simplified training environment (Figure 28). As the goal of the training was proficiency, the scripted training sequence was prolonged until the experimenter determined that no further improvement in using MVS could be obtained within a reasonable amount of training time and the participants proclaimed that they felt comfortable using the simulation.

Following the training sequence, the measured runs were performed. In order to establish a certain level of "pressure," one screen of the dual-screen setup showed MVS, the other one showed a full-screen stopwatch. The experimenter announced the current record time for the given scenario and would call out the seconds until this record during the run so as to stipulate the participant into trying to beat it. Additionally the experimenter tried to keep the participant in an conversation about the current state of the scenario in order to add an additional level of distraction. The experimenter also intervened in the case of an apparent bug in the execution of MVS.[6]

Beside these, another interruption were situations when the primary unmanned aircraft was piloted into an obstacle as neither MVS nor the default setup of the utilized GUST does prevent doing this. In order to avoid having to deal with a "loss" of a vehicle, the presence of a collision avoidance system was assumed and its function was "simulated" by the experimenter requiring the participant to return to a location prior entering the building before the mission would be continued. The time spent doing so is considered a sufficient penalty for "triggering" the collision avoidance system.

Each scenario run was followed by a brief post-scenario questionnaire (Appendix B.3) which was aimed at gathering immediate feedback with respect to the just performed run performance. The experimenter also marked the scenario and the completion time for record keeping. After completing the post-scenario questionnaire of the last run, an additional post-experiment questionnaire (Appendix B.4) was administered in order to collect some ethnographic data as well as general feedback on MVS.

[6]The observed bugs mainly caused a map view presentation which was different from the one the participant had been taught during training. An intervention normally lasted less than a two seconds and only required the experimenter to reposition a blip by a fraction so as to trigger an correcting update of the on-screen graphics.

5.3 Results

A total of 25 volunteers participated in the human subject study aimed at evaluating the performance impact of the obstacle coverage aid, completing both scenario sequences outlined in Table 2. An initial review after completion of the first stage sequence set indicated that the participants' proficiency in operating MVS improved during the course of four scenarios. The 25^{th} participant was necessary as one sequence, *S24*, was erroneously repeated before the end of the sequence set.

Appendix D presents the aggregated questionnaire data in more detail. A copy of the collected information as well as the recorded screen-capture videos is available at [14]. The statistical information given in the remainder of this chapter is either taken directly from the SDAPS report of the questionnaires (Appendix D) or from an analysis conducted in R (Appendix E).

5.3.1 Demographic

The 25 participants were on average 30 years old (A = 29.6 a, σ = 8.63 a) and predominantly identified as male (88 %, versus 12 % female). The majority (76 %) has an aerospace degree, even more (88 %) have a science, technology, engineering, or mathematics (STEM) college background, the majority were recruited graduate students from the Georgia Institute of Technology.[7] Some participants had prior experience with radio controlled flight (40 %) and a few even had experience in operating an unmanned aircraft (24 %). Although the vast majority had experiences in mouse or joystick gaming (80 % and 88 %, respectively), most of the participants had not played on a computer or a console in a long time.

More detail on the demography of the volunteers is in Appendix D.2, sections 4 to 7. A digital copy of the collected questionnaire data as well as the screen recording videos are available at [14].

[7] No monetary compensation or other awards have been granted.

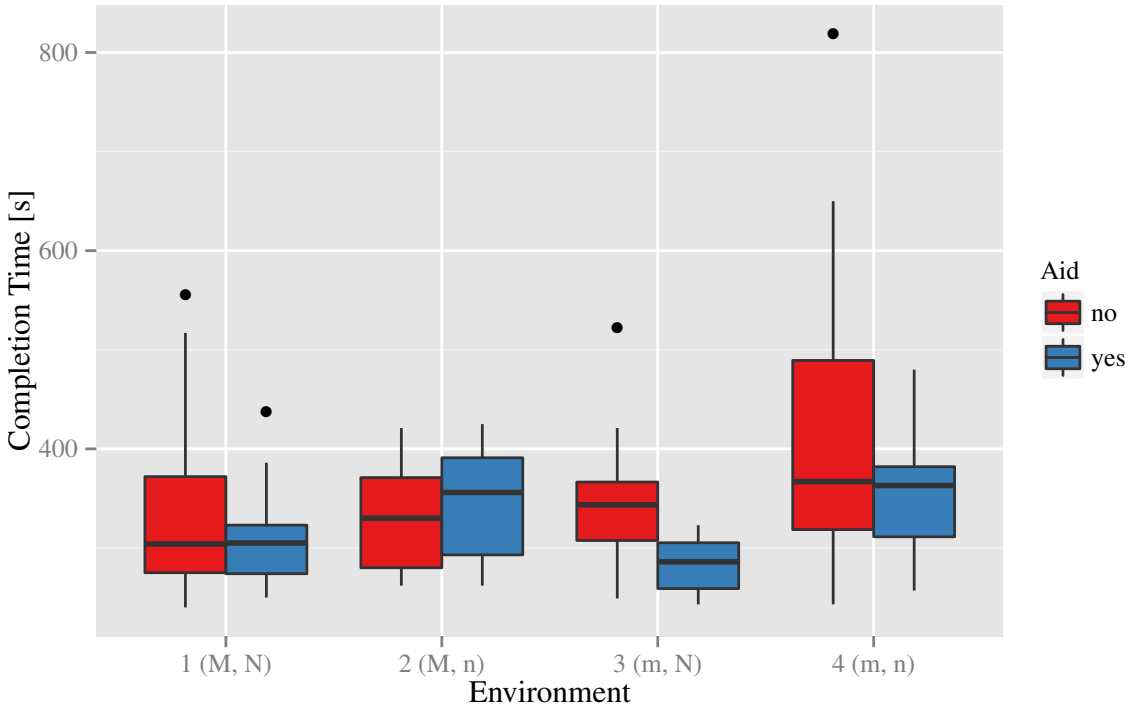


Figure 50: The completion times for the four environments, separating the runs using the obstacle coverage aid (*A*) and the ones not using it (*a*).

5.3.2 Combined Performance and Post-scenario Results

The performance of the inspection task given to the participants was measured via the time till the inspection was completed. The box-and-whisker plot in Figure 50 summarizes the descriptive statistics.[8]

Several linear mixed-effect models have been tested for adequately representing the collected data via an analysis of variance (ANOVA). The factors used to create a model of the data were taken from the controlled factors *Aid Presence*, *Map Complexity*, *Number of unmanned aircraft*, and the *Environment* factor combination; Appendix E lists the

[8] From [93]: "The upper whisker extends from the hinge to the highest value that is within 1.5 × IQR of the hinge, where IQR is the inter-quartile range, or distance between the first and third quartiles [(the 25^{th} and 75^{th} percentiles)]. The lower whisker extends from the hinge to the lowest value within 1.5 × IQR of the hinge. Data beyond the end of the whiskers are outliers and plotted as points [...]. In a notched box plot, the notches extend $\frac{1.58 \times IQR}{\sqrt{n}}$. This gives a roughly 95 % interval for comparing medians. See [64] for more details. "

individual detailed results.[9] Starting with a complete model, a statistically significant difference in the variation of the completion time was found using the factors *Aid* and *Environment* ($F(1,24) = 6.3354$, $p = 0.0189$ and $F(3,47) = 3.8915$, $p = 0.0145$, respectively.[10] The interaction between *Aid* and *Environment* was not significant.) Breaking the environment up into its factors *Number of unmanned aircraft* and *Map* and keeping all interactions shows *Aid*, *UAV*, and the single interactions with *Map* as significant, but the factor *Map* by itself does not show a statistically significant effect.[11] Eliminating the interactions with *Aid* shows a statistically significant difference difference for interaction of the factors *UAV* and *Map* ($F(1,14) = 6.3397$, $p = 0.0246$), but none for the individual factors ($F(1,33) = 4.1119$, $p = 0.0507$ and $F(1,14) = 1.2230$, $p = 0.2874$ for the number of unmanned aircraft and the map complexity, respectively).[12] Removing the interaction from the model confirms the statistical insignificance of the *Map* factor, subsequent test also show any other interaction to be of no significance.

A model only utilizing the *Aid* and *Number of unmanned aircraft* factors shows statistically significant differences for both factors ($F(1,24) = 5.7250$, $p = 0.0249$ and $F(1,33) = 4.5421$, $p = 0.0406$, respectively),[13] correlating the presence of the obstacle coverage aid (A) as well as more unmanned aircraft (N) with reductions in the completion time.

5.3.3 Observed Common Errors and Mistakes

Although not explicitly recorded, the experimenter observed several errors and mistakes that were commonly made by the participants.[14]

[9] See Appendix E.1, lines 56 through 164, and the related output in Appendix E.2, lines 2 through 170.
[10] Appendix E.2: `anova m1.2.1`, line 7 pp.
[11] Appendix E.2: `anova m2.1.1`, line 27 pp.
[12] Appendix E.2: `anova m2.1.2`, line 38 pp.
[13] Appendix E.2: `anova m3.2.2`, line 108 pp.
[14] All experiments have been recorded via screen capture, which should allow for an *a posteriori* analysis. The videos are available via [14].

5.3.3.1 Loss of Situation Awareness

The status of the communication network in MVS is indicated through the com graph whose individual edges change from green to yellow, orange, and eventually red if a particular connection is about to be disconnected due to range limitations. Aiding the understanding of the covered area is the green shading, highlighting the region currently communication-connected to the Control Station. Yet despite these graphical aids, several participants sent Vehicles to locations that did not have a working communication connection to the Control Station. In most cases the participants were able to recover from this by relocating other Vehicles, but sometimes a participant cut off all Vehicles at once by moving a disconnecting relay Node out of the coverage area of the Control Station. MVS attempts to preempt this by changing what is highlighted green, but the observed cases happened when this aid did not work as intended. If the Network is static, moving any blip changes the green shaded area so as to indicated the coverage area if that blip were cut from the network, Figure 38(f). However, the most sever cases of lost link were created during the initial setup of the network where participants would position Vehicles while others are also moving. This lead to cases where participants placed the first relay Vehicle of the Control Station just outside the Control Station's coverage area (always outlined blue) as that area was still shaded green due to the multiple other communication links from Vehicles still in the vicinity.[15]

Other cases of lost SA were observed in the scenarios with the smaller number of unmanned aircraft. In these scenarios, the coverage aid heuristic functions in the sense that it returns a result, but the result is not necessarily feasible. (Appendix C.2 has more details on these cases.) Some participants did not detect this case of brittle automation and

[15]MVS in some occasions erroneously did not generate the green shaded area correctly, mainly by ignoring the communication link of a recently moved vehicle. The experimenter noticed these situations normally very quickly and intervened manually. This seemed to happen while the participants were using the joystick control, allowing the experimenter to simply move the misbehaving blip a fraction, triggering a re-computation of the com graph and an update of the green shaded area.

seemed surprised that after the proposed formation had been realized the mission still was not immediately doable.

5.3.3.2 Pilot-induces Oscillations and Collisions

The majority of the participants did not have experience in RC flight or commanding unmanned aircraft. As such, a varying degree of proficiency in "piloting" the Vehicles through the joystick interface was to be expected. To counteract this, MVS attempts to make controlling the Vehicles easy, as stable flight is at all times maintained through the internal GNC loops of the "piloted" SVS-instance, which in the case of the presented study were GUST-controlled helicopters. While the participants issued simple velocity commands through the joystick interface, the underlying flight dynamics are not simplified (GUST has been shown to very closely simulate helicopter dynamics). As a result, particularly in close proximity to obstacles, pilot-induced oscillations were observed when participants did not make use of the proportional control of the joystick gimbals, but always seemed to push the sticks to the extremes, eliminating a gradual input and replacing it with five "fixed" ones: stick neutral and full forward, backward, left, and right.

Also attributable to not enough training with the joystick interface are errors related to yawing and strafing of the vehicle. Particularly in "nose in" situations (i.e. the Vehicle's body-x-axis does not line up with the "forward" direction of the joystick) participants sometimes got confused and yawed or translated in the wrong direction. MVS tries to support the operator's understanding of the primary unmanned aircraft's attitude by showing the body-carried coordinate system (Figure 30) and during the initial training of the participants several ways to understand the attitude have been presented. Most participants seemed to do fine as long as the Vehicle was moving, but some had trouble finding the "correct" initial joystick command when a Vehicle was hovering. As one approach to counteract "loss of control" cases demonstrated during the initial training was to simply let go of all sticks and let the Vehicle return to a hover, this led to cases where participants went

through some cycles of hover, "wrong" joystick input, sticks neutral, hover. As most of this happened during the inspection phase of the mission, several collisions between obstacles and the primary unmanned aircraft occurred.

Another source for collisions seemed to be the eagerness of some participants to complete the mission as quickly as possible. While this was certainly part of the created first-responder scenario, some participants opted to fly to fast to long (often underestimating the additional speed of the "dash" functionality, Section 4.2.3) and clipped building corners. Besides in "dash"-mode, some participants also clipped building corners during the inspection, particularly when attempting to minimize trajectory changes in response to small concavities in the inspected buildings.

5.4 Discussion

The ANOVA results support the conclusion that the presence of the provided obstacle coverage aid–as well as the number of available unmanned aircraft–reduces the completion time of the inspection task given to the participants. Additionally, the presence of the aid is more likely to have a completion time reducing effect than the use of more unmanned aircraft. As having completed the inspection quicker is assumed to be a better performance (details on the target building can be given to first-responders earlier), the proposed obstacle coverage aid is concluded to being an overall beneficial aid.

As briefly touched upon, a further improvement in how the participants interacted with MVS during the first stage of the experiment (experiment run sequences *S1* to *S12*, compare Table 2(a)) was noticed and hence, in an effort to counteract the presumably existing confounding of the presence of the aid and experience gained through testing, the second stage run sequence was used for the remaining participants. The descriptive statistics shown in Figure 50 hide the levels of the *Sequence* factor, $\{1^{st},2^{nd},3^{rd},4^{th}\}$, as the Latin square setup ensured that each environment was represented equally across the run sequences[16] and the

[16]Due to an unintentional early repetition of sequence *S24* the scenarios *A1*, *A2*, *a1*, and *a2* were indeed tested 13 times, whereas the other scenarios were tested only twelve times.

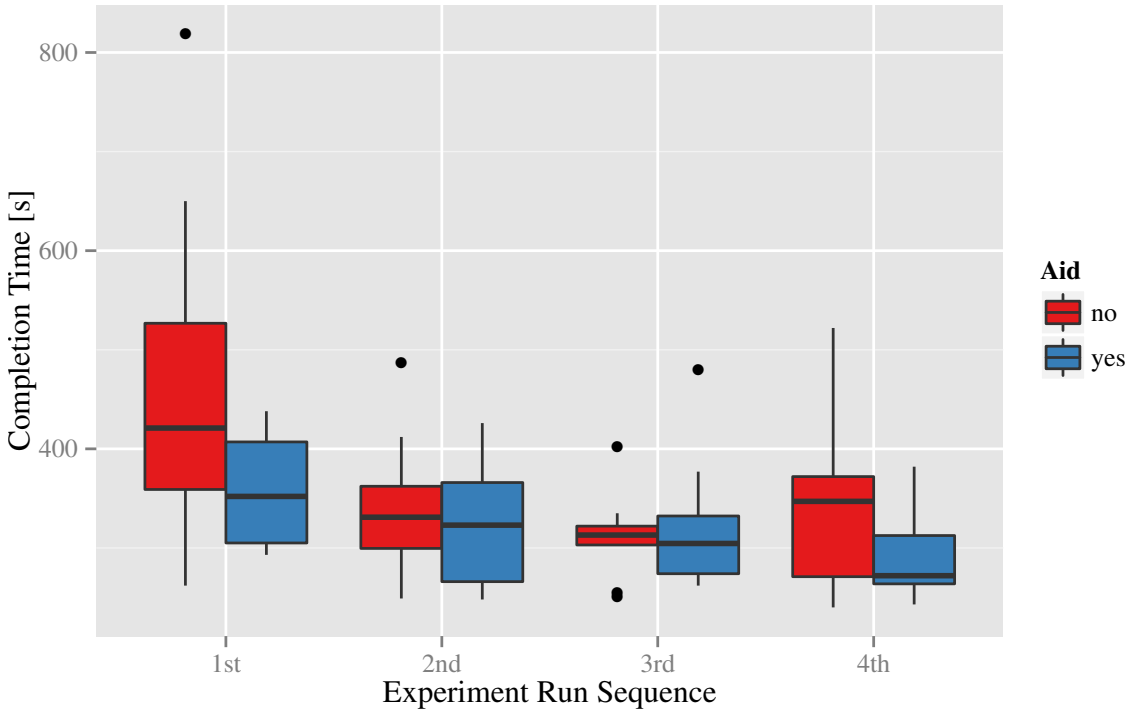


Figure 51: The completion time by experiment run. Each participant either did an $\{a, a, A, A\}$ run sequence from the first stage set (Table 2(a)) or a $\{A, A, a, a\}$ sequence from the second stage set (Table 2(b)). The four environments are uniformly distributed in each run.[16]

effect of the environment presumable is of higher interest than the effects of the run sequence. However, a descriptive statistics hiding the *Environment* inside a *Sequence* factor can be created using the same Latin square argument, as it transposes and equally uniformly distributes the environments across the sequences. Figure 51 shows this descriptive statistic and gives rise to several interesting interpretations.

As all sequences contain all environments, it would be expected that–if the sequence were not to play a role–the completion times are similar across the runs, with a slightly better performance for the runs using the aid as that has been established as being beneficial. Looking at the first run data shown in Figure 51, an initial observation is that participants starting with the aid (A) do better than those starting without the aid (a; $F(1, 23) = 4.61353$, $p = 0.0425$),[17] which matches the established expectation. Indeed, the presence of the

[17] Appendix E.2: `anova m5.6`, line 203 pp.

aid proved to be the only statistically significant factor.[18] However, the benefit apparently obtained through the aid diminishes in the second run as the participants starting without the aid seem to improve their performance considerably more than the participants starting with the aid. Comparing the improvement of the participants (the time difference between the first and the second sequence run of the same participant) who were using the aid to those not using the aid does not show a significant difference in a two-sample test ($t_W(14.062) = -1.6505$, $p = 0.0605$).[19] However, an ANOVA comparing the the completions times of all participants between their first and second runs does show a statistical difference for the participants not using the aid ($F(1, 11) = 5.8120$, $p = 0.0346$)[20] and no difference for those utilizing the aid ($F(1, 12) = 2.980$, $p = 0.1099$).[21]

Looking at the overall figure and comparing the first and second stage sequences[22] one possible explanation for the effect of the observed differences could be (mental) fatigue. The participants from the first sequence set, those starting without the aid and doing a $\{a, a, A, A\}$ sequence for their 1^{st} through 4^{th} run ({red,red,blue,blue} in Figure 51), were able to hone their proficiency during the early stages of the experiment. The variances from their second run onwards maintained fairly constant, yet the mean improved and they were able to finish the task quicker. A different picture presents itself for the participants of the second stage, i.e. those starting with the aid and finishing manually (a $\{A, A, a, a\}$ sequence for their 1^{st} through 4^{th} run, {blue,blue,red,red} in Figure 51). This group also managed to improve their mean from the first to the second run (while maintaining the variance). However, in their third run the participants were able to drastically reduce their variance, while only improving the mean a minimal amount. The fourth run then got worse

[18] Appendix E.2: `anova m5.1` through `anova m5.5`, line 172 pp.

[19] Appendix E.2: `Welch Two Sampe t-test`, line 244 pp.

[20] Appendix E.2: `anova m6.2`, line 261 pp.

[21] Appendix E.2: `anova m6.1`, line 256 pp. The improvement for the participants not starting with the aid matched the observations of the experimenter after the first sequence stage and gave reason to using the second stage instead of repeating stage one.

[22] All participants either did a first stage sequence starting without the aid, $\{a, a, A, A\}$, or they did a second stage sequence starting with the aid, $\{A, A, a, a\}$. In the context of Figure 51 this means that participants either did a {red, red, blue, blue} or a {blue, blue, red, red} sequence.

on both the mean and the variance.

The fatigue argument would mean that over the course of the experiment the participants tired to a certain degree, which, considering that the four runs took roughly 60 minutes, seems a natural assumption. Those starting without the aid could use the the early phase of the runs to improve their proficiency. The aid, available to them in their third and fourth run, then masked fatigue and the participants were able to maintain their performance and even get a little better ({red,red,blue,blue} in Figure 51). Those participants starting with the aid were able to convert the experience they gained through interacting with MVS and the aid to clamp down on the variance for their first manual, un-aided run, their third one. As the means are fairly identical for all of the second and third runs, it could be argued that this group ({blue,blue,red,red} in Figure 51) showed their overall best performance during the third run: small variance, lowest overall mean. However, when these participants had to do their fourth run without the aid, a mental fatigue might have shifted the performance to the worse: drastically increased variance and a higher mean.

The NASA Task Load Index (TLX) data collected in the post-scenario questionnaires (Appendix D.1, Section 1) seems to be in line with a fatigue argument. Figure 52 shows the data, broken down by *Sequence*, i.e. the environments and the aid presence are equally distributed in each run. Looking at the different groups, the categories for *Mental*, *Physical*, and *Temporal* demand as well as the reported level of *Frustration* could indicate the onset of fatigue towards the end of the experiment. After an initial learning (*Frustration* goes down) and improvement phase (*Performance* goes up), *Frustration* and *Mental*, *Physical*, and *Temporal* demands all seem to go up for the fourth sequence–potentially an indicator for fatigue. The self-reported *Effort* might or might not be a good indicator, as effort could be seen as being related to *Frustration* and *Performance*: due to the increased experience towards the last runs of the sequence the participants reported to have performed better[23]

[23]The self-reported increase in performance correlates with a reduction in the recorded task completion times shown in Figure 51.

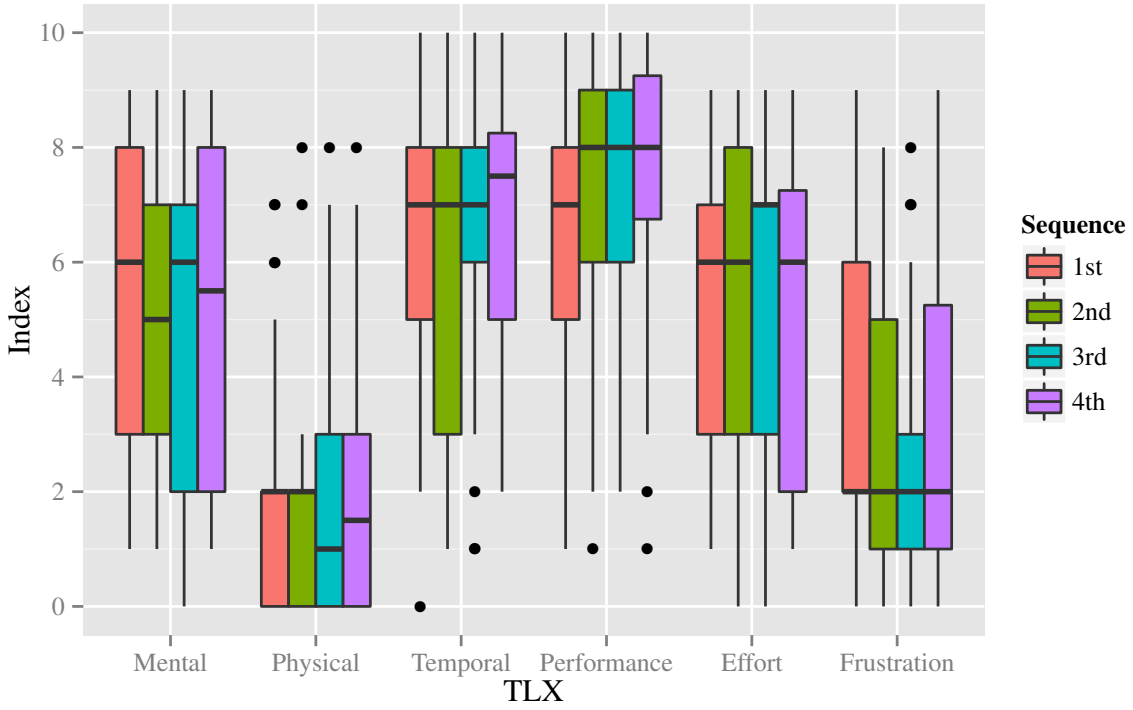


Figure 52: When ordered by experiment run sequence the self-reported NASA Task Load Index (TLX) results could indicate an improved confidence in using MVS while at the same time hinting at the onset of fatigue.

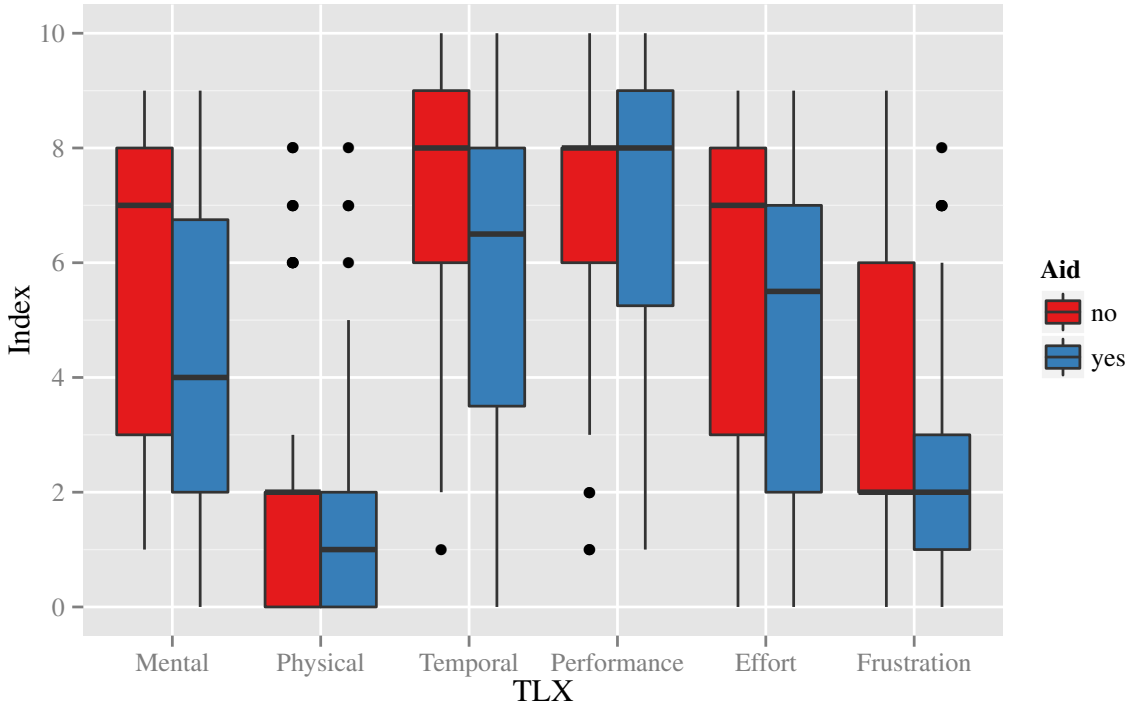


Figure 53: The availability of the coverage aid seems to generally have a positive effect on the reported NASA Task Load Index (TLX).

which could mitigate an increased level of frustration–leading to a reported effort in line with the previous runs.

Figure 53 shows the TLX data grouped by the presence of the coverage aid. In general, the self-reported data seem to indicate that the availability of the aid lead to "better" scores. *Mental*, *Physical*, and *Temporal* demand are all reported with at a lower mean, which indicates that the performed task got "easier." This is in line with the reported reduction in *Effort* and *Frustration*. The availability of the aid does not seem to have an unambiguous effect on the self-reported performance as the mean of the reported data stays the identical, yet the variance for the cases which did have the aid increases. This could be an effect resulting from the design of the experiment as the provided coverage aid does not return with a directly realizable result in two of the four environments (the automation is brittle for all cases that only have the smaller number of unmanned aircraft (n) available). The fact that the participants had to first "fix" the returned coverage result could have caused a reduced perception of performance.

In conclusion, the provided simple obstacle coverage heuristic is beneficial. The aid significantly increased the participants performance in completing the simulated first-responder task. Additionally, although not enough data is yet available to determine statistical significance, it can be argued that the aid can also support the learning of underlying principles (the second stage participants were able to perform very well on their third (their first unaided) run) as well as help to maintain a good performance even in the presence of (mental) exhaustion (the participants of the first stage sequences do not seem to show a diminished performance in the later runs). Furthermore, the collected TLX data seems to indicate that the presence of the aid had an overall beneficial effect on the reported task load.

5.5 Possible Expansions

The articles discussed in Section 2 contain many more aspects on how to optimize the interaction between the human operator and the unmanned aircraft. Of high interest for continuing work with MVS are certainly a task analysis of a UAS equipped first-responder team. Rigorously inspecting the recorded human subject study screen captures and correlating the detected errors to the results of such a study could support the efforts of improving the effectiveness of the presented coverage aid. The results could also support the creation of a first iteration of an immersive first-person video interface which could then be included in a follow-up study.

CHAPTER VI

CONCLUSIONS

The aids presented in this dissertation are only a small step towards making single-operator, multi-vehicle teams a reality. Like the underlying framework, they attempt to find a middle ground between theory, requiring several assumptions to be true, and practice, just seeking a workable solution that is "just working." Though the presented heuristics are not optimal, they are indeed good enough to be helpful. At the same time they are "theoretically practical" as they are reasonably scalable on current day computational hardware within the anticipated team and environments sizes[1] and, the limitations and requirements of the first responder application scenario always in mind, not to many corners have been cut in software. As a result, the presented MVS provides a performance improving stepping stone towards utilizing teams of unmanned aircraft in current day scenarios:

MVS is functional in the presence of imperfect maps. The framework and the aids make use of the intrinsic robust safety the Voronoi diagrams provide with respect to uncertain placement or shapes of obstacles in the environment.

MVS is usable on mobile hardware. The coverage aid as well as the overall system are usable with currently available mobile hardware: a portable computer, a gamepad controller, and a first-person-video viewing device.

MVS is connectable to COTS aircraft. The approach of MVS to wrap around the native communication API of the utilized unmanned aircraft allows the separation of aircraft, (embedded) avionics, and GNC design from mission management or command

[1]The anticipated unmanned aircraft team sizes for a first-responder scenario are in the order to 10 to 20 vehicles and in urban envrionments of maybe a square-kilometer. The conducted human subject study validated the functionality within these limits.

and control, enabling the use of different vehicles for different tasks.

MVS is deployable today. The inherent requirement for a working MANET datalink can be solved through COTS communication systems already in use in in GUST-controlled UAS, making the deployment of a MVS-controlled unmanned aircraft team an achievable possibility.

Achieving a relatively high technological readiness level has been possible through a holistic approach to developing a system helpful in a first responder scenario. An example of which is how the geometrically correct two-dimensional static polygonal environment and overall system scalability to team-sizes also dovetails with the intended use cases of MVS in imperfectly known environments, probably using mobile Control Stations. During the preparation of the human subject study, MVS has been successfully teste with teams up to 15 vehicles. Although the presented human subject study utilized a homogeneous Network of unmanned aircraft simulated by GUST, MVS aims at utilizing an inherent benefit of unmanned aircraft teams: the ability to provide a selection of unmanned aircraft with varying sizes, features, and capabilities. As this can mean using unmanned aircraft from different Single-Vehicle Systems (SVS) ecosystems, MVS only has a very small set of required features a SVS needs to provide in order to be compatible, opening up the possibility for a user to use pick unmanned aircraft from a variety of COTS systems.

A core interest of research towards enabling single-operator multi-vehicle teams is to increase the compatibility of the team comprised of the human operator and the autonomous vehicles. This requires an increased SA of the overall state of the system. The development of the MVS touched upon this in various ways, leading to the inclusion of a state machine driven framework to codify the "behavior" of Nodes. Although MVS does not provide immediate answers to the famous *What is it doing?*, *Why is it doing it?*, and *How do I change it?* questions, the framework implements an object-oriented inheritance system to alter the state machine, enabling an easy expansion of the implemented modes whilst trying

to shield non-software-development researchers from the underlying framework code.

6.1 Contributions

The described work presents a collection of various smaller contributions which in conjunction form the provided MVS.

In the context of the MVS framework (Chapter 3), contributions are primarily related to the software architecture of MVS:

- Provision of a software framework to combine several single vehicle UAS into a multi-vehicle system under an open source license.
- Implementing the commercially available data distribution system RTI Connext to use MVS across several host computers connected via internet protocol.
- Provision of a path planning aid to compute maximum clearance routes through a know static polygonal environment, based upon a geometrically correct Voronoi partitioning from the commercial software VRONI.
- Utilizing the open source library VisiLibity to provide a method to simulate range-limited line-of-sight communications in a static polygonal environment, comparable to those utilized in MANET protocols.
- Structuring the available source code under the PIMPL coding scheme to simplify version compatibility and to provide an API between research users and research developers of MVS.

In the context of the GUI and the provided aid algorithms (Chapter 4), contributions are related to the described aid heuristics:

- Provisions to generate a geometrically correct vertex-reduced polygonal sub-environment as a localized environment for computationally intensive algorithms.

- Provision of a fast heuristic to generate a range-limited line-of-sight multi-hop RF connection between a location in the environment and a target obstacle.
- Provision of a heuristic providing range-limited line-of-sight RF coverage in the vicinity of an obstacle, roughly related to the "fortress problem with guards in the plane."
- Provision of a GUI enabling a single operator to use multiple unmanned aircraft to conduct an inspection of a target beyond LOS and single-hop RF distance.

In the context of the conducted human subject study (Chapter 5) the contribution relates to validating the benefits of the provided systems:

- Confirming the performance enhancing benefits of the the provided aid algorithms in a scenario comparable to the envisioned first responder use case.

In the hopes to simplify the reuse of the available code beyond the work related to this dissertation, the available sources also contain a large amount of Doxygen-formatted comments as well as an example on how to integrate MathWorks' MATLAB Engine to interface MVS with an instance of MATLAB.

The sources for MVS are available at [15] and the collected data of the human subject study including the screen capture videos are available for download at [14].

6.2 Review and Outlook

The approach to create the MVS framework with the goal of developing a deployable software is a main contributor to the extensive time requirements needed for the software development of MVS. Resulting from this, the "tangible" elements of MVS, the GUI and the aids available to an operator, do not stand out as the major part of the presented work, but as an equal element.

However, with the framework in place, MVS provides the potential to extend the "tangible" elements through coverage aids that are distributed, adaptive to operator interventions, and better integrated into the state machine governed mode logic. Reviewing the informal observations of mistakes made during the human subject study, these yet-to-be-implemented advanced aids could help to overcome the repeated mistakes of unintentional Network disconnections, either through moving a Node not realizing that is its a separating vertex or through deploying all Vehicles to the inspection site, not realizing that the immediate com graph neighbor of the Control Station also has been dragged outside the single-hop RF range of the Control Station.

Another interesting problem that was uncovered during the human subject study also, in a broader sense, relates to HMI: operator training. In the context of the motivating scenario, MVS can be used to simulate deployments of UAS-aided first-responder teams and, for example, explore where hangars of unmanned aircraft could be placed in a metro area to reduce response times. The statistical setup of the human subject study stipulated training all participants to proficiency in using MVS and the conducted training sessions were used to showcase various manual approaches to complete the inspection task and demonstrated the use and shortcomings of the provided aid. Despite successfully completing the training, the participants considerably improved their performance through the course of the four conducted timed experiments, a fact indicating that not only do autonomous systems need to improve cooperation with human operators, but also that human operators need to be better prepared to collaborate with autonomous systems. MVS could be used as an aid to improve how training in this field is conduced.

APPENDIX A

SOFTWARE SOURCE CODE AND LICENSES

Besides the standard C and C++ libraries, MVS utilizes third party software available under the following licenses:

Apache Xerces	Apache License 2.0
Boost	Boost Software License 1.0
CodeSynthesis XSD	GPL v2 or commercial
Eigen	MPL v2
GUST	proprietay (no public distribution of sources)
Qt	LGPL v2.1, LGPL v3, GPL v3 or commercial
RTI Connext DDS Professional	commercial (free for academic use)
SDL	zlib license
VisiLibity	LGPL v3
VRONI	commercial (free for academic use)

The sources of MVS itself (roughly 20 000 source lines of code), and the related wrappers for third-party libraries are released under differing open-source licenses at `https://github.com/mvsframework`:

MVS	Apache License 2.0
AMG	Apache License 2.0
Vroni Wrapper	LGPL v3
VisiLibity Wrapper	LGPL v3

The sources of MVS are to a large extend documented in the Doxygen standard and the build environment provides a build target to create or update the HTML or LaTeX Doxygen documentation.

APPENDIX B

HUMAN SUBJECT STUDY FORMS

B.1 Experimenter Cheat Sheet

The following script was used by the experimenter as a guide through the human subject study. The sections *1 Preparations* and *2 Rundown* deal with administrative elements to be done before the practical training (described in *4 Training*) were to start. The section *3 Motivational Scenario* was read to the participants verbatim to explain their role as the dedicated UAS operator in the first responder scenario.

Experimenter Cheat Sheet

1 Preparations

- Use wired mouse
- Disable screen-saver
- Enable visualization of mouse clicks (Meta+∗)
- Have some water to drink
- Prepare (stamped) consent forms
- Prepare questionnaires (4 post-scenarios, 1 post-experiment) and a pen
- Prepare screen recorder
- Prepare a Stopwatch

2 Rundown

1. Welcome the participant and thank the participant for volunteering in this research study
2. Inform the participant about the necessary first step: obtaining informed consent.
   (a) Explain that all research conducted at Georgia Tech is overseen by a review board, the IRB (Institutional Review Board) and note that this is to ensure that research is conducted ethically and all involved know their rights.
   (b) Hand the participant the 'Participant Consent' form and ask her or him to read through the document. Point out that they should ask about anything they feel they don't understand fully. (Let the participant read through the document.)
3. Ensure the participant is 'informed' (after the participant has finished reading.)
   (a) State that all this, including this first step, is completely voluntary and that the participant can always stop participating and leave at any time, without having to give any reason.
   (b) Ask about any question the participant might have about the form
   (c) Go through the requirements list ('5. Inclusion/Exclusion Criteria) and ensure that the participant qualifies to participate in the study. (If nor, thank her or him for the effort and state that they, unfortunately, cannot participate.)
   (d) Go through the 'Procedures' section and explain the steps
   (e) Talk about the foreseeable risks (section 7), the benefits of the study, and compensation (i.e. the participants will not receive any compensation).
   (f) Reiterate on the participants rights, section 14.
4. Obtain consent:
   (a) Explain that the name of the participant will not be collected and that all collected data will be anonymized by a participant ID.
   (b) Explain that part of this is that the participant doesn't actually have to sign the 'Participant Consent' form, but that you are now verbally asking whether the participant has understood that form and that the

participant agrees to volunteer in this research study. If the participant gives verbal informed consent, continue. If not, thank the participant for volunteering her or his time and explain that unfortunately she or he cannot participate in the experiment. If, and only if, verbal consent was given, continue with the explanation of the simulation setup.

3 Motivational Scenario

Read the following to the participant:

"The fire department of A Town has upgraded their technology and added Unmanned Aerial Vehicles (UAVs) to their First Responder Squads throughout the city.

You are a dedicated UAV operator attached to Fire Station 42 and you and your squad have just been called to a high rise fire in downtown. During the ingress to your command station near the fire, you have loaded a map of the environment into your system, determined your operational altitude and obtained the outlines of the buildings you cannot fly over, i.e. you have a map of obstacles at the ready.

While your colleagues are getting the equipment of the trucks and ready, you are tasked to gather some vital intelligence on the fire in the 13th floor of the highrise in order to better direct the imminent rescue mission, i.e. through prioritizing which rooms to try to get to first. Your task is to get a look from the outside of the building at the burning floor so that you can quickly find and identify people trapped in their rooms.

It is of the utmost importance that you gather that data as quickly as possible as your squad is waiting on you to determine where to go first."

4 Training

Explain that–as time is of the essence–the actual runs will be timed and the performance of the participant will be determined by the measured time required to finish the building scanning task. As such, the following training is important as all participants should reach the same level of proficiency in interacting with the simulator.

4.1 Boot Camp

- Use the dedicated `training` map.
- Right side: 2D map; Left side: Browser data about nodes
- Explain visuals in the map
  - Operational boundary (red border)
  - Obstacles (colored polygons)
  - "You": Blip in the center of the blue outlined R/F polygon
  - Blip iconography: outline roughly 3 m (rotor diameter), tail boom is back
  - R/F polygon (gray/green): coverage area
  - Route graph: maximum clearance paths through environment
- Explain the basic interface
  - Pan (click and drag map), Zoom (scroll wheel or `+`/`-` keys)
  - Blip movement: drag and drop blip (note tool tip, real time R/F polygon prediction, colored outline matches Blip main color; ESC cancels a started drag-and-drop operation)
  - Blips use route graph to navigate; exception when relocating to a place that is close (approx. three rotor diameters)
- Explain context menu
  - Name of Blip and various Blip-specific visuals
  - Join/Leave Commands (skip for now)
  - Note that context menu is also available in browser
- Explain the browser

  - Demonstrate resizing, opening of sub-branches
  - Point out mode descriptor, and main color of Blip
- Explain the "network" concept
  - Join the GCS and point out the changes (Blip changes from gray to color, R/F polygon changes from gray to green)
  - Point out the mode changes
- Add two extra nodes
  - Point out color, name and mode in browser
  - Point out COM graph
  - "Join" them (one via browser, one via map context menu [note extra entries])
  - Point out magenta waypoints (only N next ones are shown)
  - Point out final waypoint with matching color.
  - Point out how their R/F area turns green
- Explain the coverage and connectedness notion
  - build a connection to the other side of Rectangle 1 (one Blip to the North, one to the East of it.)
  - point out the yellow velocity vector
  - move one node to the max. R/F range and point out the changes in color in the COM graph
  - drag and drop the Blips and point out the adaption of the green R/F covered area
- Briefly show the View menu and the possibility to disable the route and COM graphs
- Explain joystick interface (incl. dash-mode)
  - Enable via context menu
  - Note FOV polygon and the body frame (can be disabled vie the context menu)
  - Note how the green R/F coverage area only shows areas that provide connection to the GCS.
  - Demonstrate the mode 2 joystick interface
  - Demonstrate the dash mode
- End with micro-scenario (scan the little building, Square 4)
  - Inspect all sides of the building, fly so close that the tip of the red arrow is inside the building

4.2 Advanced Training

- Also use the `training` map.
- Add two more nodes (for a total of four)
- Introduce different tactics
  - coverage
  - depth-first
  - distributed
- Explain aid
  - setting a target
  - proposing formation
  - realizing formation
- Demonstrate shortcomings of aid

4.3 Achieve Proficiency

Allow participants to play in the `mout` or `training` scenario until they feel comfortable

B.2 Consent Form

The consent form for the human subject study as approved by the Internal Review Board (IRB) of the Georgia Institute of Technology.

The signed originals of all participants are on file with the principal investigators.

Participant Consent

1 Project Title

Single Operator Control of Autonomous Multi-Vehicle Swarms

2 Investigators

Principal Investigators: Dr. Eric N. Johnson, eric.johnson@gatech.edu, (404) 385-2519 and Dr. Karen Feigh, karen.feigh@gatech.edu, (404) 385-7686

Student Investigators: Claus Christmann, hcc@gatech.edu, (404) 894-0657

3 Protocol Title

Single Operator Control Interface Study for a Swarm of Unmanned Aerial Vehicles

4 Purpose

This project aims at investigating the situational awareness (SA) of a human operator interacting with a swarm of autonomously acting unmanned aerial vehicles (UAVs) through a computer based real-time simulation.

For this research, you are being asked take the role of the operator of a swarm of UAVs. As the operator, you are tasked with completing a simulated mission mimicking a first responder scenario in which you can utilize a swarm of UAVs to explore the scene of a high rise fire. Your goal during the mission is to collect information which you could then hand off the the other first responders on site in order to aid their actions. In order to study various aspects of your interaction with the swarm as well as your understanding of the overall simulated situation, you are asked to perform this mission in several slightly different scenarios.

5 Inclusion and Exclusion Criteria

Participants in this study need to

- be familiar with the general functionality of graphical user interfaces commonly found in computer applications,
- be familiar with mouse-based computer interaction,
- be familiar with gamepad-style human-machine-interfaces,
- have a BS degree (or equivalent),
- be proficient in English,
- be 18 years of age or older.

Additionally, subjects who do not speak English or who might not be able to give consent cannot participate.

6 Procedures

The experiment will follow the outline below:

1. An **introductory briefing** will seek informed consent, explain the goal of the experiment, how this relates to you performing several simulated mission scenarios, and give you a rough initial overview of the utilized simulator.

2. A sequence of **tutorials** will be conducted with you during which the use of the simulator is further explained and clarified. The tutorials conclude once you feel comfortable operating the simulator and have experienced the fundamental problems posed during the simulated mission.

3. During the actual **experiment**, you will conduct several missions. The missions will temporarily be paused and you will be given a short questionnaire evaluation your current situational awareness. After each scenario, you will also be asked to fill out a brief post-scenario questionnaire.

4. A **debriefing** will happen after the conclusion of all mission scenarios. You will be asked for an overall feedback related to your experience using the simulator.

The entire procedure is expected to last about two (2) hours. You can request a break at any time and you are also free to stop participating at any time.

7 Foreseeable Risks and Discomforts

Every study involves some risk, however, this study is considered to have low risk. There is the possibility of discomfort and fatigue, comparable to the level you would experience during day to day computer work.

8 Benefits

There are no direct benefits to you for participating in this study. However, this study may help increase the understanding of human-computer-interaction between a single operator and multiple autonomous agents, which, among others, has a potential benefit to future first responders.

9 Compensation and Costs

You are neither compensated for participating in this study nor are there any costs for you resulting from your participation.

10 Confidentiality

All information that you give during the study will be handled confidentially. Personal information about you will not be published or made available to any third party in any form. Your name will neither be collected nor be attached to or affiliated with any of the data. The investigators will be careful to ensure that no identifiable patterns in your actions and questionnaire responses (including demographic information such as your experience) are released in a way that would allow for any outsider to guess your identity. All raw data from this experiment will be stored in a locked facility on the Georgia Tech campus and all electronic information will be kept on password-protected GT computer infrastructure.

Once the analysis and documentation of this experiment are complete, electronic and paper

stores of results will be archived in a locked facility within the principal investigators' Georgia Tech office or laboratory.

To make sure that this research is being carried out in the proper way, the Georgia Institute of Technologies Institutional Review Board (IRB) has reviewed the study procedures and will review study records. Additionally, the Office of Human Research Protections (OHRP) may also look at study records.

11 Injury or Adverse Reactions

Reports of injury or reaction should be made to the Principal Investigator of this study. Neither the Georgia Institute of Technology nor the principal investigator has made provision for payment of costs associated with any injury resulting from participation in this study.

12 Contact

If you have questions about the research, call or write Dr. Eric Johnson at (404) 385-2519, Montgomery Knight Building, Room 415-2, Georgia Institute of Technology, 270 Ferst Drive, Atlanta GA 30332-0150.

13 Voluntary Participation and Withdrawal

You are participating in this study voluntarily and have the right to withdraw from it at any time without penalty.

14 Participants Rights

1. Your participation in this study is voluntary. You do not have to be in this study if you don't want to be.
2. You have the right to change your mind and leave the study at any time without giving any reason and without penalty.
3. Any new information that may make you change your mind about being in this study will be given to you.
4. You will be given a copy of this consent form to keep.
5. You do not waive any of your legal rights by signing this consent form.

If you have any questions about your rights as a research participant, you may contact:

Ms. Melanie Clark
Office of Research Integrity Assurance
Georgia Institute of Technology
(404) 894-6942

Your signature below indicates that the researchers have answered all of your questions to your satisfaction, and that you consent to volunteer for this study.

If you sign below, it means that you have read (or have had read to you) the information given in this consent form, and you would like to be a volunteer in this study.

_____
Participant Signature

_____
Participant Name Date

B.3 Post-scenario Questionnaire

An example of the post-scenario questionnaire the participants filled out after each experiment run. Section *4 Performance* was filled out by the experimenter to note the scenario and the completion time.

The questionnaires, generated with Scripts for Data Acquisition with Paper-based Surveys (SDAPS) [7], provide unique quick-response codes which were used to keep track of the questionnaires a participant answered. The post-scenario questionnaires were read electronically and the automatically extracted results were manually corrected and verified.

Single Operator Control of Autonomous Multi-Vehicle Swarms
Post-Scenario Questionnaire

This questionnaire is automatically read by a computer program. Please use a pen for filling in your answers.

Check: ☒ You can check any number of boxes in selection questions.

Uncheck to correct: ■ For questions with a range (1–5) choose the answer the mark that fits best.

1 NASA Task Load Index (TLX)

1.1 **Mental Demand**: How mentally demanding was the task?

Very Low (0) ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ (10) Very High

1.2 **Physical Demand**: How physically demanding was the task?

Very Low (0) ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ (10) Very High

1.3 **Temporal Demand**: How hurried or rushed was the pace of the task?

Very Low (0) ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ (10) Very High

1.4 **Performance**: How successful were you in accomplishing what you were asked to do?

Perfect (10) ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ (0) Failure

1.5 **Effort**: How hard did you have to work to accomplish your level of performance?

Very Low (0) ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ (10) Very High

1.6 **Frustration**: How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low (0) ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ (10) Very High

2 UAV Swarm

2.1 **Network Size**: How would you rate the number of UAVs that were available to you?

☐ too little ☐ about right ☐ more than necessary

2.2 Would you think that more UAVs would have made your task easier?

☐ Yes ☐ No

2.3 Which statement would you rather agree with?

☐ I was mainly busy managing my swarm. ☐ I was mainly busy operating my primary UAV.

3 Mission Review

3.1 **Strategy**: Which of the following statements would best describe your initial approach to your mission?

☐ Thinking about R/F coverage first and then start placing UAVs.
☐ Starting to deploy UAVs and then watch how the coverage works out.

3.2 **Tactic**: In your recollection, what would best describe the tactic you used to finish your task?

☐ (1) Place secondary UAVs to create R/F coverage to and around the target and then (2) use a *yet unused* UAV as the primary vehicle to complete the inspection task.
☐ (1) Select a primary UAV and move it towards the target and (2) deploy secondary UAV when necessary to increase the R/F coverage from the GCS to the target.
☐ (1) Build up R/F coverage to and around the target and then (2) use *already deployed* UAVs one after the other as the primary UAV to inspect the target.
☐ **Other**: Please describe briefly how you approached your mission in the textbox in question 3.4.

3.3 **Hindsight**: Would you say that the final R/F coverage of your secondary UAVs turned out to be roughly how you anticipated it to be before you started moving UAVs?

☐ Agreed: Yes, the secondary UAVs roughly provided coverage where I initially thought it would end up being.
☐ Disagreed: The R/F coverage provided by the secondary UAVs didn't end up matching my expectations.
☐ Not Applicable: I didn't think about where and how the UAVs would provide coverage, I just started moving them and reacted to how the coverage was build up.

3.4 If you answered *Other* in question 3.2 please describe briefly (and legibly) how you approached your mission.

4 Performance

4.1 Scenario

☐ a1 ☐ a2 ☐ a3 ☐ a4 ☐ A1 ☐ A2 ☐ A3 ☐ A4 Time:

B.4 Post-experiment Questionnaire

An example of the post-experiment questionnaire the participants filled out after the last experiment run.

The questionnaires, generated with Scripts for Data Acquisition with Paper-based Surveys (SDAPS) [7], provide unique quick-response codes which were used to keep track of the questionnaires a participant answered. The post-experiment questionnaires were read electronically and the automatically extracted results were manually corrected and verified. The entries in the free form text boxes were manually transcribed.

Single Operator Control of Autonomous Multi-Vehicle Swarms
Post-Experiment Questionnaire

This questionnaire is automatically read by a computer program. Please use a pen for filling in your answers.

Check: ☒ You can check any number of boxes in selection questions.
Uncheck to correct: ■ For questions with a range (1–5) choose the answer the mark that fits best.

1 Placement Aid

1.1 Helpfulness: How helpful was presence of the placement aid?

☐ Very Helpful ☐ Helpful ☐ Neutral ☐ Distracting ☐ Very Distracting

1.2 How would characterize the placement aid's behavior?

☐ Predictable
☐ Unexpected, but understandable in hindsight.
☐ Unexpected, and even in hindsight rather odd.
☐ Random

1.3 Looking back, if you would have the chance to use the aid in scenarios where you weren't given the option, would you have used it?

☐ Yes ☐ No

1.4 Do you think that the placement aid had a positive impact on your task performance?

☐ Yes, I believe the aid helped me perform better. ☐ No, I think the aid hindered my performance. ☐ I am not sure, I don't think it helped or hindered me.

1.5 Aid Purpose: How would you describe your expectation towards the placement aid's purpose on a scale from *faster results*, which might not be optimal, to *better results*, which could be slower to compute?

Faster Results ☐ ☐ ☐ ☐ ☐ Better Results

2 Visual Interface

2.1 Looking back at the various visual elements of the interface, how would you rate the following items?

	Very Helpful	Helpful	Neutral	Distracting	Very Distracting
Gray Maximum Clearance Paths	☐	☐	☐	☐	☐
Green R/F Coverage Area	☐	☐	☐	☐	☐
Outlined R/F Coverage Prediction	☐	☐	☐	☐	☐
Black Field-of-View Indicator	☐	☐	☐	☐	☐
COM Graph Indicator	☐	☐	☐	☐	☐

2.2 Overall, how would you rate the visual presentation of all that data?

☐ Very Helpful ☐ Helpful ☐ Neutral ☐ Distracting ☐ Very Distracting

3 Piloting Interface

3.1 Which physical interface would you prefer to pilot your primary UAV?

☐ R/C Radio (the type you used)
☐ Gampepad-style Controller (comparable to a Playstation or Xbox controller)
☐ Mouse and Keyboard (WSAD)
☐ Realistic Helicopter Replica (Cyclic Joystick, Collective/Throttle Lever, and Rudder Peddals

4 Background

4.1 **Demographics**:

Age: _____ Gender Identity: _____

4.2 **Education**: What is your current student status or your highest degree obtained?

☐ Undergraduate ☐ Graduate ☐ Doctoral ☐ Post-Doctoral

4.3 **Field**: In which field did or will you get this degree?
Please write down the name of your field and categorize it as either being a part of STEM (Science, Technology, Engineering, Mathematics) or not.

Name: _____ ☐ STEM ☐ not STEM

4.4 **Training**: Have you had any sort of military or law enforcement training?

☐ No ☐ Yes

4.5 **Tactical Lead**: If you answered *Yes* to military or law enforcement training above, have you directly led a small unit in a tactical situation in training or active duty/combat?

☐ No ☐ Yes ☐ N/A (answered *No* in question 4.4)

5 UAV/UAS Operator Experience

For the purpose of the questions in sections 5 and 6 the differentiation of UAS operation and R/C flight is by the capability of the controlling interface to guide the air vehicle through means other than direct manipulation of (aerodynamic) control surfaces.
Example: The simulation you just used would qualify as UAS operation as were able to control the aircraft through drag and drop with the mouse. A system that only provides an interface comparable to the joystick control of the primary UAV (i.e. no map overview and/or waypoint control) would only qualify for R/C flight operation (independently of the possible level of input augmentation/stabilization).

5.1 **UAS Operations**: Do you have any experience in operating Unmanned Aerial Vehicles/Systems?

☐ No ☐ Yes

If you answered *No*, please continue at question 6.1. If you answered *Yes*, please also answer the following questions.

5.2 **Operator Role**: What was your primary role as an operator?

☐ Pilot ☐ Payload/Systems Other: _____

5.3 **Experience**: How much total time have you accumulated (in any role) as a UAS operator?

Hours: _____

6 Radio Controlled Flight Experience

For the purpose of the questions in sections 5 and 6 the differentiation of UAS operation and R/C flight is by the capability of the controlling interface to guide the air vehicle through means other than direct manipulation of (aerodynamic) control surfaces.
Example: The simulation you just used would qualify as UAS operation as were able to control the aircraft through drag and drop with the mouse. A system that only provides an interface comparable to the joystick control of the primary UAV (i.e. no map overview and/or waypoint control) would only qualify for R/C flight operation (independently of the possible level of input augmentation/stabilization).

6.1 **R/C Flying**: Do you have any experience in radio controlled flight?

☐ No ☐ Yes

If you answered *No*, please continue at question 7.1. If you answered *Yes*, please also answer the following questions.

6.2 How often do you fly radio controlled aircraft on average?

☐ daily ☐ a few times a week ☐ a few times a month ☐ a few times a year
☐ at most once a year

6.3 When did you actually fly radio controlled aircraft the last time?

☐ today ☐ this week ☐ last week ☐ this month ☐ last month
☐ I actually haven't flown in a long time

6.4 Which type of transmitter configuration are you used to?

☐ Mode 1 ☐ Mode 2 ☐ Mode 3 ☐ Mode 4

6.5 How would you rate your experience/comfort/skill in radio controlled flight?

☐ Novice/Beginner ☐ Casual ☐ Average ☐ Proficient ☐ Expert

7 Video Game Experience

7.1 **Mouse Gaming**: Do you have any experience with mouse-based real time strategy games?
(Examples: Warcraft, Total Annihilation, Command and Conquer, Starcraft, Age of Empires)

☐ No ☐ Yes

If you answered *No*, please continue at question 7.5. If you answered *Yes*, please also answer the following questions.

7.2 How often do you play mouse-based strategy games on average?

☐ daily ☐ a few times a week ☐ a few times a month ☐ a few times a year
☐ at most once a year

7.3 When did you actually play a mouse-based strategy game the last time?

☐ today ☐ this week ☐ last week ☐ this month ☐ last month
☐ I actually haven't played in a long time

7.4 How would you rate your experience/comfort/skill using a mouse as a game interface?

☐ Novice/Beginner ☐ Casual ☐ Average ☐ Proficient ☐ Expert

7.5 **Joystick Gaming**: Do you have any experience with gamepad or joystick based games?

☐ No ☐ Yes

If you answered *No*, please continue at question 4.4. If you answered *Yes*, please also answer the following questions.

7.6 How often do you play gamepad-based games on average?

☐ daily ☐ a few times a week ☐ a few times a month ☐ a few times a year
☐ at most once a year

7.7 When did you actually play a gamebad-based game the last time?

☐ today ☐ this week ☐ last week ☐ this month ☐ last month
☐ I actually haven't played in a long time

7.8 How would you rate your experience/comfort/skill using a gamepad-style controller as a game interface?

☐ Novice/Beginner ☐ Casual ☐ Average ☐ Proficient ☐ Expert

8 General Feedback

8.1 If you could add pre-programmed behaviors, what would you add?

☐ "Go to Building" ☐ "Build R/F Cover" ☐ "Follow me" ☐ "Inspect Building"

other:

8.2 Which part of the system did you have the most trouble with?

8.3 Which part of the system would you consider the most helpful?

8.4 How would you expand the system's capabilities? Are there features that you have been missing?

Intentionally left blank.

APPENDIX C

EXPERIMENT SETUP

C.1 Latin Squares

The MATLAB function below is based upon the algorithm presented at [10]. For more details and examples of Latin squares in various forms see [90].

Listing C.1: MATLAB function computing a row-complete Latin square in reduced form.

```
function L = rowCompleteLatinSquare(n)
%% ROWCOMPLETELATINSQUARE Compute a row-complete Latin square of order n.
% Note: n must be even and greater than zero.
%
% This function follows
% http://personal.maths.surrey.ac.uk/st/H.Bruin/MMath/LatinSquares.html
% to which I have been pointed to from
% https://math.stackexchange.com/q/991078
%

if n<=0 || mod(n,2)~=0
error('n must be even and greater than zero.');
end

% create difference vector
d = 1:n-1;
for i= 2:2:(n-1)
d(i)=n-i;
end
clear i;

% preallocate memory
L = zeros(n,n);

% initialize first column
L(:,1) = (0:n-1)';

% populate Latin square
for i=2:n
```

```
    L(:,i) = mod(L(:,i-1)-d(i-1),n);
end

% normalize (match order in first row and first column)
L = L(L(1,:)+1,:);

% reduce form (remap the symbols so that they are in natural order)
L_reduced = zeros(n,n);
for i=0:n-1
  L_reduced(L==L(i+1,1))=i;
end
L = L_reduced;
clear L_reduced i;

end
```

This function is also available via [20].

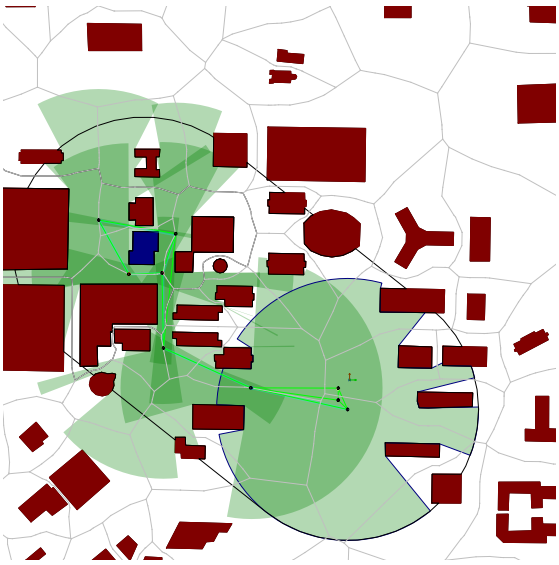
C.2 Scenario Solutions

The presented coverage aid is able to generate a solution proposal for all of the environments, depicted in Figure 54. The proposed solutions for the environments *1* and *3*, those using the higher number of unmanned aircraft, are directly usable as the number of Vehicles required to realize the proposed solution is smaller than the number available. For the environments *2* and *4* this is not the case and the participants had to alter the proposed solution to accomplish the inspection mission.

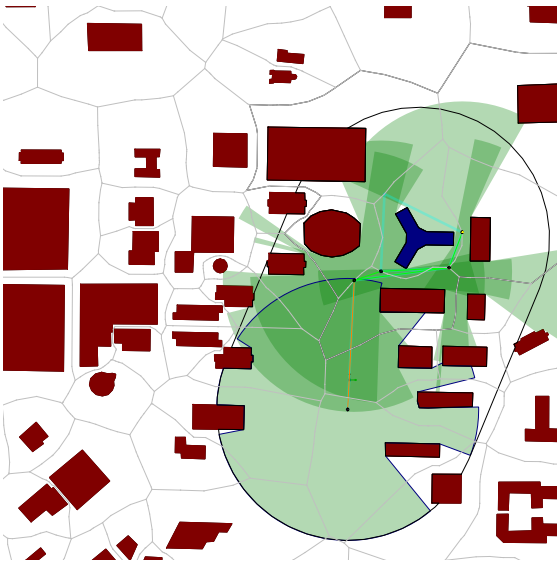
For environment *2* solving this problem is presumably simpler than for environment *4*, as in the former one of the Vehicles position on the "loop" part of the solution is redundant (the blue blip in the South-West corner) and can be used to scan the building, only requiring some careful piloting a the northern face where the coverage has a small gap. For environment *4* approach is not possible as the gap resulting from using one of the loop-assigned blips is too large to scan the building. A solution either requires a different positioning of the blips or an approach that doesn't rely on complete coverage of the related obstacle cell.

However, all environments are "solvable" via a coverage approach, i.e. there were

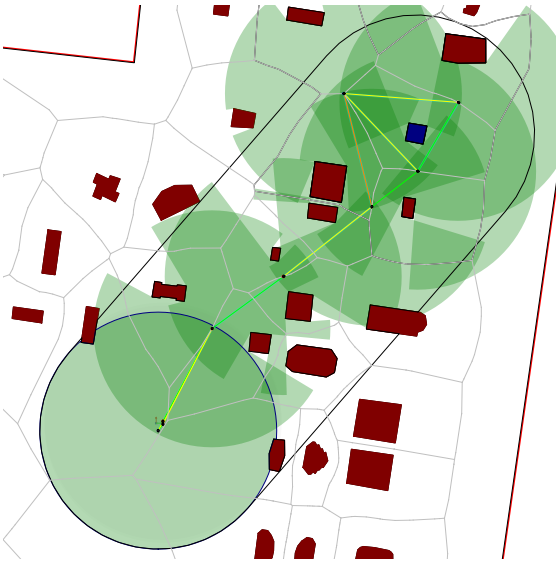
enough Vehicles available in each scenario to generate full Network coverage of the obstacle cell of the target building and have one Vehicle left to perform the inspection task. Figure 55 shows a possible coverage "solution" for each of the environments.



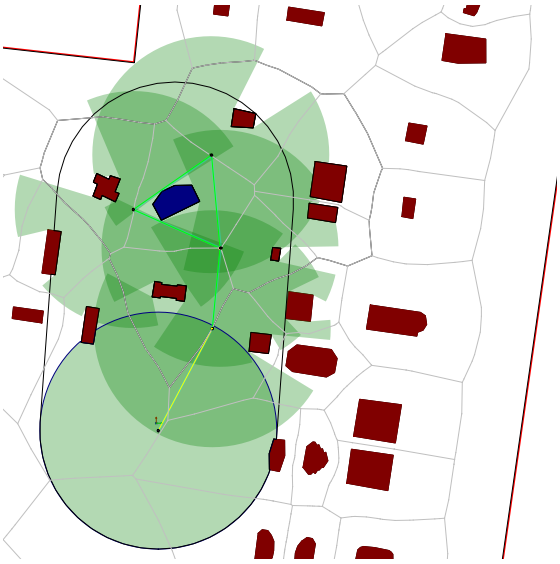
(a) Environment 1 (M, N): The proposed solution requires 6 of the 8 available Vehicles to cover the target building envelope.



(b) Environment 2 (M, n): The proposed solution would require 5 Vehicles for coverage, but only 4 are available.

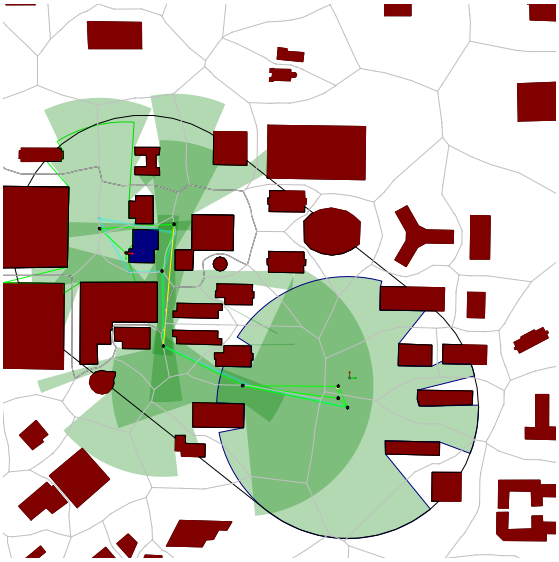


(c) Environment 3 (m, N): The proposed solution requires 6 of the 8 available Vehicles to cover the target building envelope.

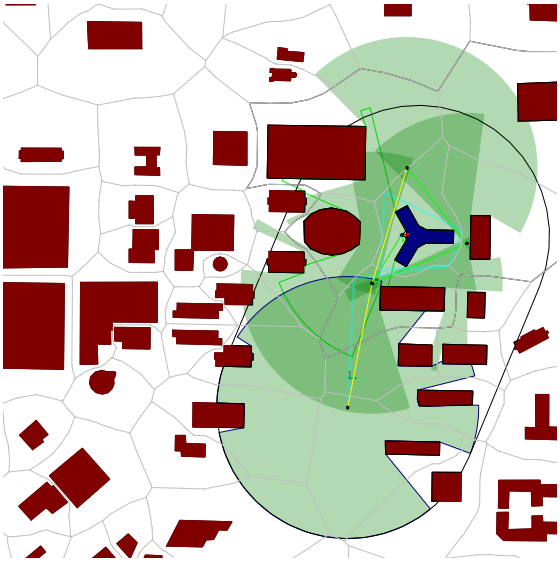


(d) Environment 4 (m, n): The proposed solution requires all 4 of the 4 available Vehicles to cover the target building envelope.

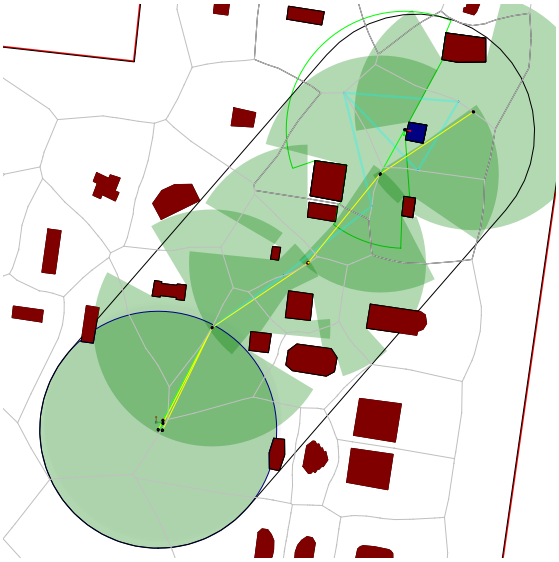
Figure 54: The solution proposals generated by the obstacle coverage aid work in the case of the scenarios with the higher number of unmanned aircraft (N), but create infeasible solutions in the scenarios with fewer unmanned aircraft (n). The proposed solutions are only always feasible when the number of available Vehicles is at least one more than the number required for coverage.



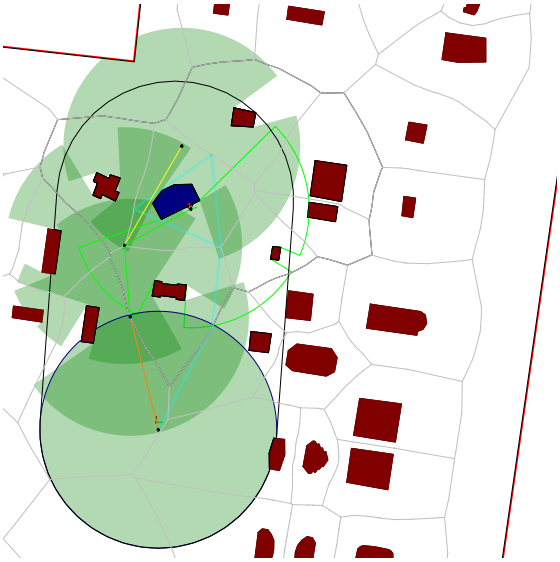
(a) Environment 1 (M, N): The quadrilateral coverage allows to use the Vehicle in the NE or SW corner without compromising coverage



(b) Environment 2 (M, n): It is possible to create full coverage with only 3 Vehicles, freeing the 4^{th} one up for the inspection task.



(c) Environment 3 (m, N): For this detached square target complete coverage is always possible with only 2 Vehicles.



(d) Environment 4 (m, n): For this larger polygon coverage with only 2 Vehicles can be achieved only on one side due to the limited number of relays.

Figure 55: Manually placing the Vehicles can reduce the number of required Vehicles (N scenarios with more unmanned aircraft) or enable the use of a complete coverage approach (n scenarios with less unmanned aircraft).

APPENDIX D

COLLECTED QUESTIONNAIRE DATA

The following reports summarizing the data collected through the post-scenario and post-experiment questionnaires were created based on the SDAPS [7] report sources.

D.1 Post-scenario Questionnaire Data

A total of 100 questionnaires from 25 participants were collected. However, one questionnaire happened to only contain the scenario and the completion time, i.e. no responses from the participant were collected for that run. For this reason, some responses explicitly list the number of received answers as 99 instead of the expected 100.

SDAPS starts scales at 1 and counts up, independently of the labels written to the axis. As a result, numeric values such as a mean are potentially not in accordance with an associated graphical representation. (An example for this are the NASA TLX reports.)

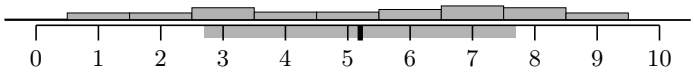
1 NASA Task Load Index (TLX)

1.1 **Mental Demand**: How mentally demanding was the task?

Very Low (0) – (10) Very High

1 % 8 % 8 % 14 % 9 % 9 % 12 % 16 % 14 % 8 % 0 %

Answers: 99
Mean: 6.2
Standard-Deviation: 2.5

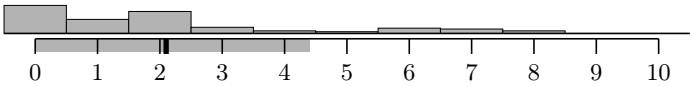


1.2 **Physical Demand**: How physically demanding was the task?

Very Low (0) – (10) Very High

32 % 16 % 25 % 7 % 3 % 2 % 6 % 5 % 3 % 0 % 0 %

Answers: 99
Mean: 3.1
Standard-Deviation: 2.3

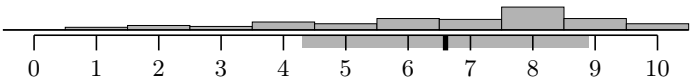


1.3 **Temporal Demand**: How hurried or rushed was the pace of the task?

Very Low (0) – (10) Very High

0 % 3 % 5 % 4 % 9 % 7 % 13 % 12 % 26 % 13 % 7 %

Answers: 99
Mean: 7.6
Standard-Deviation: 2.3

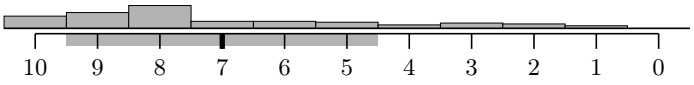


1.4 **Performance**: How successful were you in accomplishing what you were asked to do?

Perfect (10) – (0) Failure

14 % 18 % 26 % 8 % 8 % 7 % 4 % 6 % 5 % 3 % 0 %

Answers: 99
Mean: 4.0
Standard-Deviation: 2.5

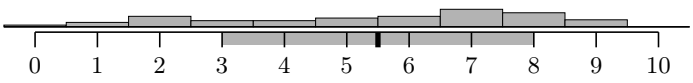


1.5 **Effort**: How hard did you have to work to accomplish your level of performance?

Very Low (0) – (10) Very High

2 % 5 % 12 % 7 % 7 % 10 % 12 % 20 % 16 % 8 % 0 %

Answers: 99
Mean: 6.5
Standard-Deviation: 2.5

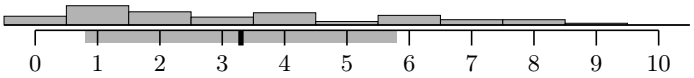


1.6 **Frustration**: How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low (0) – (10) Very High

10 % 22 % 15 % 9 % 14 % 4 % 11 % 6 % 6 % 2 % 0 %

Answers: 99
Mean: 4.3
Standard-Deviation: 2.5



2 UAV Swarm

2.1 **Network Size**: How would you rate the number of UAVs that were available to you?

too little	14 %	
about right	47 %	
more than necessary	38 %	

2.2 Would you think that more UAVs would have made your task easier?

Yes	46 %	
No	53 %	

2.3 Which statement would you rather agree with?

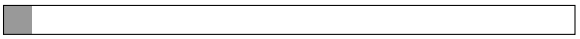
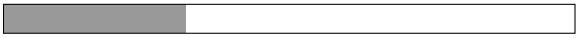
I was mainly busy managing my swarm.	27 %	
I was mainly busy operating my primary UAV.	72 %	

3 Mission Review

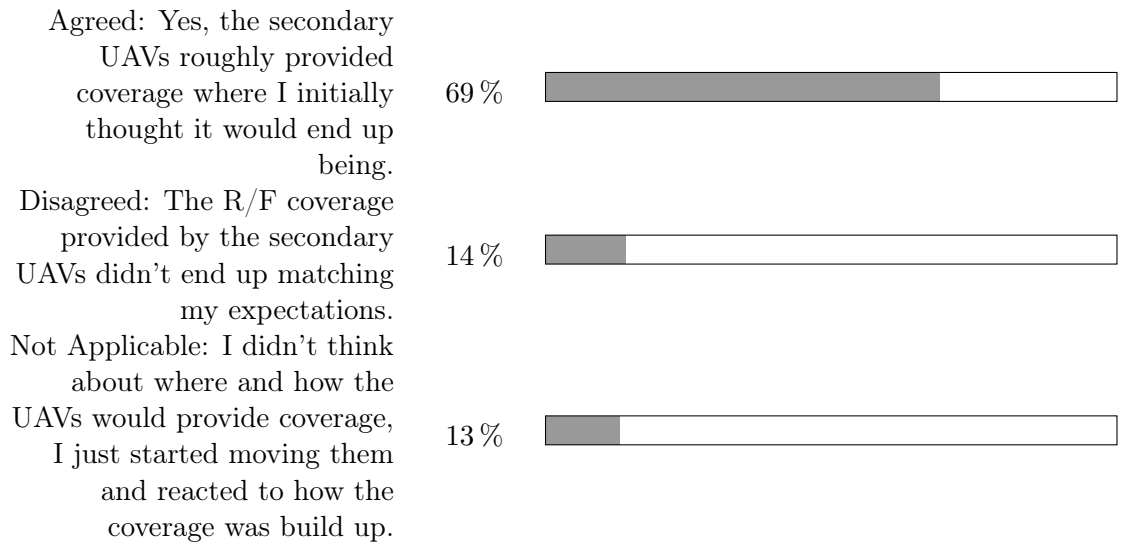
3.1 **Strategy**: Which of the following statements would best describe your initial approach to your mission?

Thinking about R/F coverage first and then start placing UAVs.	51 %	
Starting to deploy UAVs and then watch how the coverage works out.	42 %	

3.2 **Tactic**: In your recollection, what would best describe the tactic you used to finish your task?

(1) Place secondary UAVs to create R/F coverage to and around the target and then (2) use a *yet unused* UAV as the primary vehicle to complete the inspection task.	54 % 
(1) Select a primary UAV and move it towards the target and (2) deploy secondary UAV when necessary to increase the R/F coverage from the GCS to the target.	5 % 
(1) Build up R/F coverage to and around the target and then (2) use *already deployed* UAVs one after the other as the primary UAV to inspect the target.	32 % 
Other: Please describe briefly how you approached your mission in the textbox in question 3.4.	9 % 

3.3 **Hindsight**: Would you say that the final R/F coverage of your secondary UAVs turned out to be roughly how you anticipated it to be before you started moving UAVs?



3.4 If you answered *Other* in question 3.2 please describe briefly (and legibly) how you approached your mission.

used the aid

---

like 3rd answer, but manually flew only one UAV

Used the aid to find an initial guess to get the swarm moving, then made adjustments to coverage. Selected a craft which was redundant to the comm network to become primary search

---

Used the aid to get an initial solution to the coverage problem; however, the proposed formation had a communication link through a very narrow gap – decided to chose a ¨more robust¨ path-but still used the suggested waypoints around the building, with slight adjustments.

I let the system decide on placement, then directed redundant UAVs to the nodes furthest from my ground station. I then used this added security to le me navigate a different unassigned UAV without having to worry any longer about R/F coverage.

Same as 442. [I let the system decide on placement, then directed redundant UAVs to the nodes furthest from my ground station. I then used this added security to le me navigate a different unassigned UAV without having to worry any longer about R/F coverage.]

I started by placing UAVs one by one, startin with a node closest to the ground station. I tried using as few UAVs as possible, I then used unassigned UAVs as redundant nodes closest to the building. I used the last unused UAV to fly around the building knowing that I would have no coverage issues due to previsou steps.

See survey 142. [I started by placing UAVs one by one, startin with a node closest to the ground station. I tried using as few UAVs as possible, I then used unassigned UAVs as redundant nodes closest to the building. I used the last unused UAV to fly around the building knowing that I would have no coverage issues due to previsou steps.]

Tried to gain some time with manually controlling a secondary UAV (sprint). When reverting back to primary UAV for manual control I lost track of which UAV I used as primary UAV...

4 Performance

Manual summary of the free form **Time** data for scenario **a1**:

Answers: 13
Mean: 341
Standard-Deviation: 98.4

Manual summary of the free form **Time** data for scenario **a2**:

Answers: 13
Mean: 333
Standard-Deviation: 54.6

Manual summary of the free form **Time** data for scenario **a3**:

Answers: 12
Mean: 346
Standard-Deviation: 73.9

Manual summary of the free form **Time** data for scenario **a4**:

Answers: 12
Mean: 429
Standard-Deviation: 164.9

Manual summary of the free form **Time** data for scenario **A1**:

Answers: 13
Mean: 311
Standard-Deviation: 53.7

Manual summary of the free form **Time** data for scenario **A2**:

Answers: 13
Mean: 343
Standard-Deviation: 58.5

Manual summary of the free form **Time** data for scenario **A3**:

Answers: 12
Mean: 283
Standard-Deviation: 28.3

Manual summary of the free form **Time** data for scenario **A4**:

Answers: 12
Mean: 354.8
Standard-Deviation: 63.4

D.2 Post-experiment Questionnaire Data

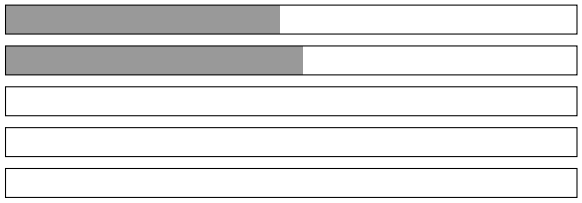
The post-experiment questionnaire contained some fields into which the participants could enter text freely in order to not require the collected responses to be binned. Some of the collected responses could, however, be binned after the collection. These manually processed evaluations were possible for gender identity, age, field of study, and operator experience. The corresponding sections are labeled as "manual summary."

Note that sections 5 and 6 allowed the participants to skip any but the first question, yet the percentages given are still with respect to all collected questionnaires.

1 Placement Aid

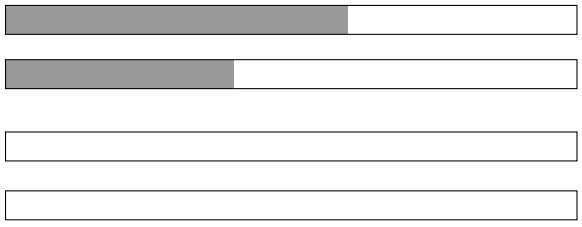
1.1 Helpfulness: How helpful was presence of the placement aid?

Very Helpful	48 %
Helpful	52 %
Neutral	0 %
Distracting	0 %
Very Distracting	0 %



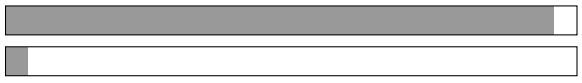
1.2 How would characterize the placement aid's behavior?

Predictable	60 %
Unexpected, but understandable in hindsight.	40 %
Unexpected, and even in hindsight rather odd.	0 %
Random	0 %



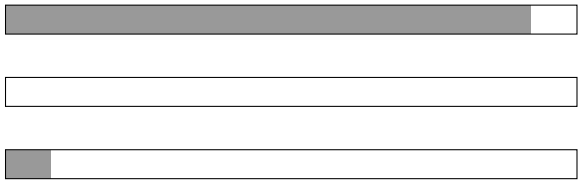
1.3 Looking back, if you would have the chance to use the aid in scenarios where you weren't given the option, would you have used it?

Yes	96 %
No	4 %



1.4 Do you think that the placement aid had a positive impact on your task performance?

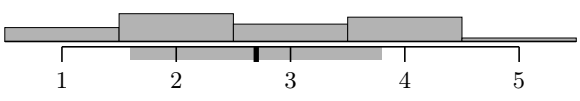
Yes, I believe the aid helped me perform better.	92 %
No, I think the aid hindered my performance.	0 %
I am not sure, I don't think it helped or hindered me.	8 %



1.5 Aid Purpose: How would you describe your expectation towards the placement aid's purpose on a scale from *faster results*, which might not be optimal, to *better results*, which could be slower to compute?

Faster Results – Better Results	16 %	32 %	20 %	28 %	4 %

Answers: 25
Mean: 2.7
Standard-Deviation: 1.1

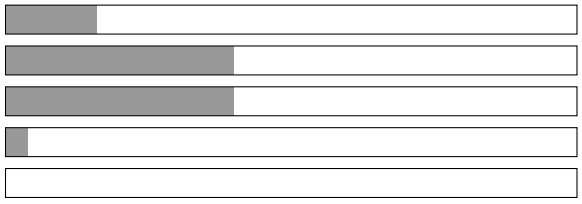


2 Visual Interface

2.1 Looking back at the various visual elements of the interface, how would you rate the following items?

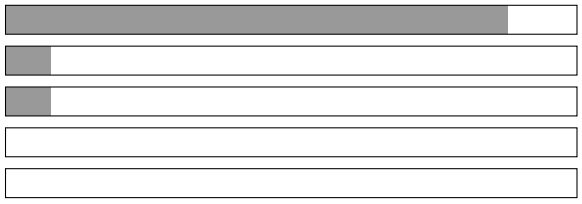
2.1.1 Gray Maximum Clearance Paths

Very Helpful	16 %
Helpful	40 %
Neutral	40 %
Distracting	4 %
Very Distracting	0 %



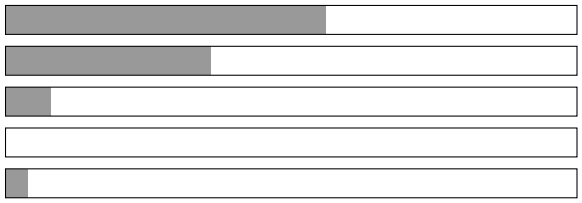
2.1.2 Green R/F Coverage Area

Very Helpful	88 %
Helpful	8 %
Neutral	8 %
Distracting	0 %
Very Distracting	0 %



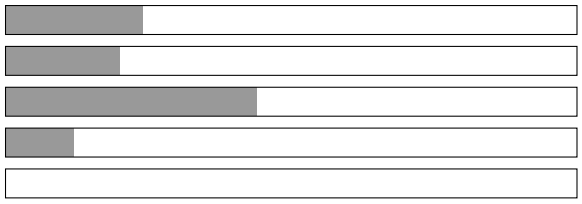
2.1.3 Outlined R/F Coverage Prediction

Very Helpful	56 %
Helpful	36 %
Neutral	8 %
Distracting	0 %
Very Distracting	4 %



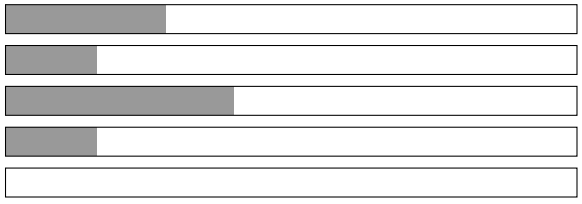
2.1.4 Black Field-of-View Indicator

Very Helpful	24 %
Helpful	20 %
Neutral	44 %
Distracting	12 %
Very Distracting	0 %



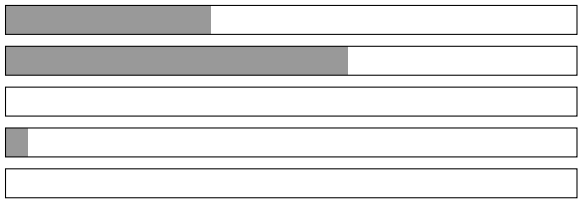
2.1.5 COM Graph Indicator

Very Helpful	28 %
Helpful	16 %
Neutral	40 %
Distracting	16 %
Very Distracting	0 %



2.2 Overall, how would you rate the visual presentation of all that data?

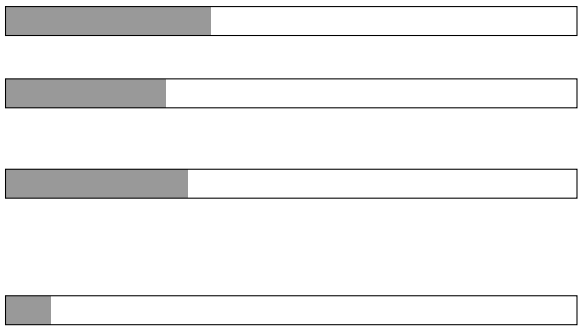
Very Helpful	36 %
Helpful	60 %
Neutral	0 %
Distracting	4 %
Very Distracting	0 %



3 Piloting Interface

3.1 Which physical interface would you prefer to pilot your primary UAV?

R/C Radio (the type you used)	36 %
Mouse and Keyboard (WSAD)	28 %
Gampepad-style Controller (comparable to a Playstation or Xbox controller)	32 %
Realistic Helicopter Replica (Cyclic Joystick, Collective/Throttle Lever, and Rudder Peddals	8 %



4 Background

4.1 **Demographics**:

Age:	100 %
Gender Identity:	100 %

29

M

29

Male
36
male
23
Male
24
M
31
m
24
m
24
male
24
male
36
Male
24
Female
24
Male
44
M
23
Male
58
Male
28
female
25
Male
23
Male
22
M
31
female
35
Man
26
Male
22

Male
43
male
32
Male

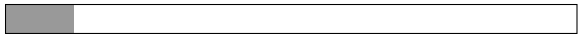
Manual summary of the free form **Gender Identity** data from above:

Female	12 %	
Male	88 %	

Manual summary of the free form **Age** data from above:

Answers: 25
Mean: 29.6
Standard-Deviation: 8.63

4.2 **Education**: What is your current student status or your highest degree obtained?

Undergraduate	12 %	
Graduate	52 %	
Doctoral	32 %	
Post-Doctoral	4 %	

4.3 **Field**: In which field did or will you get this degree?
Please write down the name of your field and categorize it as either being a part of STEM (Science, Technology, Engineering, Mathematics) or not.

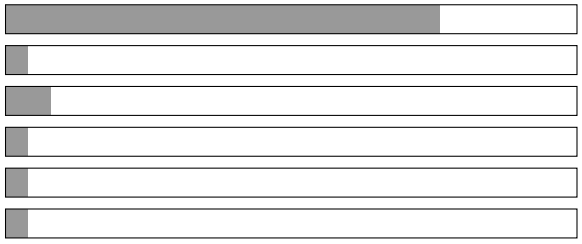
Name:	100 %	
STEM	88 %	
not STEM	12 %	

Aero Eng
Aerospace
Aerospace Eng.
Aerosapce Engineering
Aerospace
AE
Aerospace Eng.
aerospace
Aerospace Engineering
Aerospace Eng

Aerospace Eng.
AE
Engineering
Mechanical and Aerospace Engineering
Engineering
Aerospace Eng
Undergrad: Physics & Maths; Grad: Aerospace Engineering
Aerospace Engineering, Control & Simulation
Industrial Engineering
Communications
Aerospace
Aerospace Engineering
Aerospace Eng.
Physical Therapy
Fine Arts

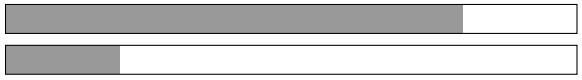
Manual summary of the free form **Name** data from above:

Aerospace Engineering	76 %
Communications	4 %
Engineering	8 %
Fine Arts	4 %
Industrial Engineering	4 %
Physical Therapy	4 %



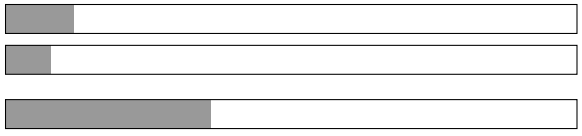
4.4 **Training**: Have you had any sort of military or law enforcement training?

No	80 %
Yes	20 %



4.5 **Tactical Lead**: If you answered *Yes* to military or law enforcement training above, have you directly led a small unit in a tactical situation in training or active duty/combat?

No	12 %
Yes	8 %
N/A (answered *No* in question 4.4)	36 %

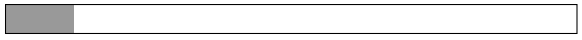
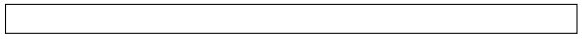


5 UAV/UAS Operator Experience

5.1 **UAS Operations**: Do you have any experience in operating Unmanned Aerial Vehicles/Systems?

No	76 %	
Yes	24 %	

5.2 **Operator Role**: What was your primary role as an operator?

Pilot	20 %	
Payload/Systems	12 %	
Other:	0 %	

5.3 **Experience**: How much total time have you accumulated (in any role) as a UAS operator?

Hours:	24 %	

5 _____
1 _____
40 _____
< 5 _____
300 _____
3 _____

Manual summary of the free form **Hours** data from above:

Answers: 6
Mean: 59
Standard-Deviation: 119

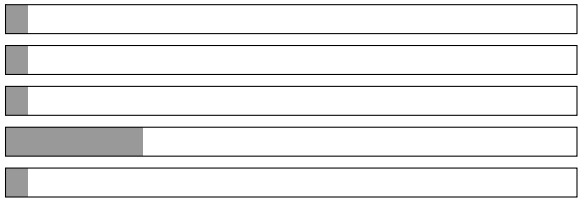
6 Radio Controlled Flight Experience

6.1 **R/C Flying**: Do you have any experience in radio controlled flight?

No	60 %	
Yes	40 %	

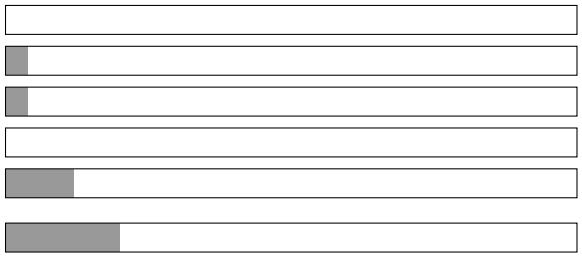
6.2 How often do you fly radio controlled aircraft on average?

daily	4 %
a few times a week	4 %
a few times a month	4 %
a few times a year	24 %
at most once a year	4 %



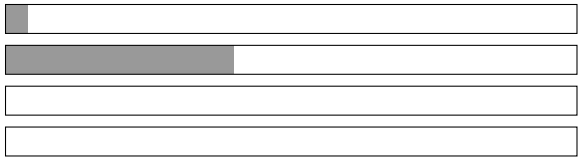
6.3 When did you actually fly radio controlled aircraft the last time?

today	0 %
this week	4 %
last week	4 %
this month	0 %
last month	12 %
I actually haven't flown in a long time	20 %



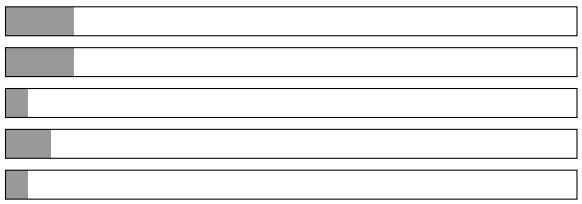
6.4 Which type of transmitter configuration are you used to?

Mode 1	4 %
Mode 2	40 %
Mode 3	0 %
Mode 4	0 %



6.5 How would you rate your experience/comfort/skill in radio controlled flight?

Novice/Beginner	12 %
Casual	12 %
Average	4 %
Proficient	8 %
Expert	4 %

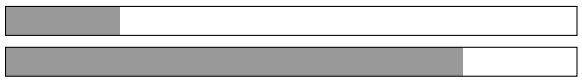


7 Video Game Experience

7.1 **Mouse Gaming**: Do you have any experience with mouse-based real time strategy games?

(Examples: Warcraft, Total Annihilation, Command and Conquer, Starcraft, Age of Empires)

No	20 %
Yes	80 %

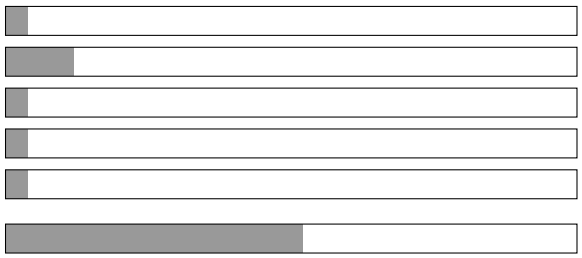


7.2 How often do you play mouse-based strategy games on average?

daily	12 %
a few times a week	8 %
a few times a month	12 %
a few times a year	36 %
at most once a year	12 %

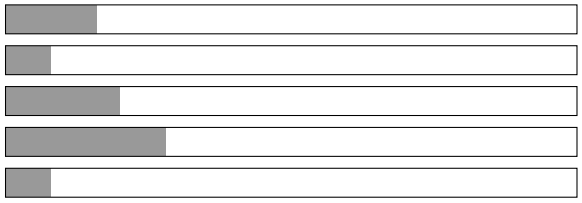
7.3 When did you actually play a mouse-based strategy game the last time?

today	4 %
this week	12 %
last week	4 %
this month	4 %
last month	4 %
I actually haven't played in a long time	52 %



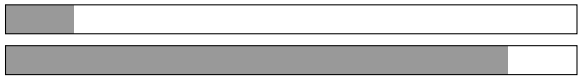
7.4 How would you rate your experience/comfort/skill using a mouse as a game interface?

Novice/Beginner	16 %
Casual	8 %
Average	20 %
Proficient	28 %
Expert	8 %



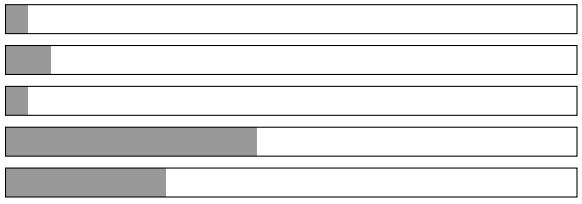
7.5 **Joystick Gaming**: Do you have any experience with gamepad or joystick based games?

No	12 %
Yes	88 %



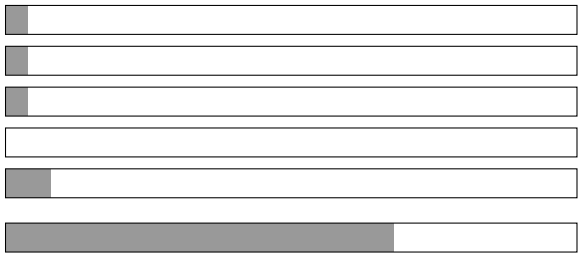
7.6 How often do you play gamepad-based games on average?

daily	4 %
a few times a week	8 %
a few times a month	4 %
a few times a year	44 %
at most once a year	28 %



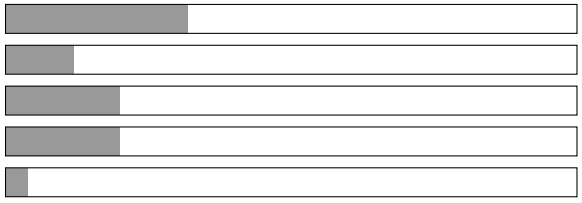
7.7 When did you actually play a gamebad-based game the last time?

today	4 %
this week	4 %
last week	4 %
this month	0 %
last month	8 %
I actually haven't played in a long time	68 %



7.8 How would you rate your experience/comfort/skill using a gamepad-style controller as a game interface?

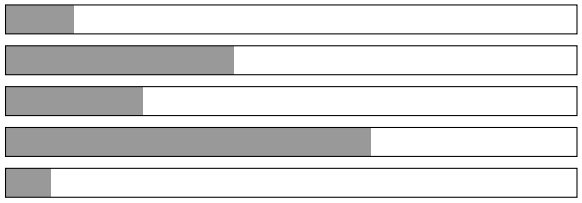
Novice/Beginner	32 %
Casual	12 %
Average	20 %
Proficient	20 %
Expert	4 %



8 General Feedback

8.1 If you could add pre-programmed behaviors, what would you add?

"Go to Building"	12 %
"Build R/F Cover"	40 %
"Follow me"	24 %
"Inspect Building"	64 %
other:	8 %



1) Emergency Stop (individual, all) 2) Auto-Joing and form towards target (all) _____
Draw a manual path for a UAV _____

8.2 Which part of the system did you have the most trouble with?

keeping trafic of aircraft in close proximity of each other and 'grabbing' the right one
Piloting
figuring out the expected R/F coverage of multiple agents
1) All visuals in the same screen (be able to decide on available info [I would remove: comgraph, gray-lines, purple waypoints in between]) 2) Ability to move vehicles ¨out of range¨ withough warning 3) Dealing with too many vehicles
Estimating R/F coverage
stress level caused me to ignore graphics and I lost link in mission 2
1) Setting up coverage w/out aid 2) inspecting building 3) Remembering to ¨join¨ before taking initial action
temporarily losing R/F coverage while placing UAVs
1) R/F coverage area (shaded green) 2) Vehicle Marker Color (hard to distinguish)
With a large number of agents, the viewfans became easy to mix up
Maneuverung the UAV manually was the most stressful & time consuming
1) Knowing whether the ¨proposed formation¨ was robust 2) The proposed formation didn't account for my goal of moving 1 UAV around the whole building 3) Inspecting the building
Corresponding vehicles on map with browser
Accidentally losing coverage while moving ¨waystation¨ UAVs.
R/F cover
Navigating in body axis
When vehicles were autonomously navigating, it was difficult to tell if a vehcile was ¨loitering¨ because it was where it wanted to be, or because I hadn't told it to do anything yet.
The UAVs following the gray seperation lines, this sometimes led unpredictable loss of R/F coverage. Also, UAVs are a bit slow when they are not in manual. mode.
Flying the UAV around the buildings
1) Piloting - overadjusting on turns as I inspected the building 2) Accounting for the time it would take to move the aircraft to adjust coverage
controlling joystick
flying with the R/C radio
Controlling the helicopter using joystick
Initial strategizing
Maintaining signals between drones while repositioning

8.3 Which part of the system would you consider the most helpful?

The ¨aid¨
Green R/F circles
the aid
1) the provided ¨aid¨ 2) green areas

Inventarisation and Quick initial formation
The aide.
1) Aid! 2) turbo charge feature 3) RF polygons
the aid
Formation Realization
The viewfasn & comm graph made the solution pretty intuitive – just build a chain that connects all the nodes to the ground.
the green shadows & the red/yellow indicator bars
1) Drag and drop capability is very helpful 2) The Level of shading is a good heuristic for knowing where UAV's can be placed. 3) The dark outline of the R/F when a UAV was ¨picked up¨ was essential to maximizing coverage of the buildings.
green coverage areas
The RF coverage and comm. links maps/graphs
The Aid
green coverage areas to position UAVs. It helped me figure out the constraints on my UAV placement.
Green indication of R/F coverage and the darker/lighter shades associated with redundant coverage.
1) Green Circles 2) Aid
1) The aid was extremele helpful in gettin gstarted - not having to analyse coverage right away, but then having ability to get started and adjust on the fly. 2) As far as function, the drag-and-drop feature from the menu bar was extremely helpful when it was difficult to distinguish one aircraft from the next
aids function to specify appropriate position to get full communication coverage
The green areas, very helpful to know the boundaries
Predictions of the RF coverage before the vehicle is actually placed at the candidate location.
R/F Coverage Maps
The outlined R/F coverage prediction map was most helpful.

8.4 How would you expand the system's capabilities? Are there features that you have been missing?

1) Enable aircraft to fly off route graph when in auto mode (safely!) 2) Hold coninuous ¨strafe¨ 3) Allow forward flihgt while surveying (moving sensor)
auto-fly the building inspection and let operator worry about swarm only
1) warning on the expected out-of-range possibilities for specific agent while deploying 2) provision of more definite (expected) R/F covergage
collision controller
Flag if extra UAV's are available
Just what is in 8.1 but it is doable without

1) Magic ¨join-all¨ button 2) autonomous inspection

automatic ¨stay in bounds¨ function when flying manually

1) ¨Select All¨ to join vehicles 2) FPV camera

An improvement on the palnning aid would be the ability to adjust the proposed formation with checks for coverage & comm graph. Would have been a bit more effective if I could make the small adjustmens, but the command 'realize formation' with a single click, rather than placing acFt individually

1) I really would have liked the ¨inspect building¨ feature. 2) I also would have liked to set constraints for the aid algorithm. For instance, with the square building to tell the aid to only consider 3 points of contact would have been useful

1) Have the ¨proposed form.¨ have an option to leave 1 UAV out & not use the paths. 2) Have a robustness measure on the proposed formation _or_ setting to say I don't want small thresholds for R/F connections

Completely automated mode

A system to automatically scan along a given face of a building. I could be (re)building RF coverage while the scan was automatic.

1) Dual rate. Adjustment of the stick sensitivity 2) Less ambiguous icon.

The predicted coverage lines were somtimes hard to see – it made predicting the coverage and/or expected performance difficult. Maybe make those lines bolder.

The RF Line (not the shaded regions) which indicate where RF coverage WILL be were difficult to see. It would have been helpful to a quick toggle between current (shaded) RF coverage and a ¨future¨ RF coverage, which might be in a different color. Having both at the same time was confusing.

Proposed direction for scanning the building (i.e. starting from the back. CW or CWW).

The green circles - areas of coverage were extremely helpful but the colors were sometimes hard to decipher. SOme of the lines were faint and seemed to create ¨white noise¨.

Autonomous UAV relay to keep maintaining the communication relay with the ground control station.

Collision Avoidance; I ran into many buildings

1) The field of view and route map lines have the same colour. I make field of view a yellow region (yellow because of light bulb color). 2) It was difficult to identify UAVs by the colored annular disk. I would give a name identification on the map on top of the vehicles. 3) The joystick was extrememly sensitive to my inputs, I would reduce the control gains.

Follow Me Mode

A ¨follow me¨ feature would be very helpful.

APPENDIX E

ANALYSIS OF VARIANCE (ANOVA)

E.1 R Script

The script file used to generate the statistical analysis of the human subject study data.

Listing E.1: R script to run ANOVA analysis in R.

```
## SETUP ----------------------------------------------------------------------

# clean up
rm(list=ls())

# loading libraries
library("nlme")
library(reshape2)


## DATA IMPORT ----------------------------------------------------------------

# Importing the performance relevant data from the post-sceanario questionnaires
#
# "Questionnaire"    : ID of the post-scenario questionnaire
# "Participant"       : ID of the participant
# "Sequence"           : A counter indicating the sequence or order in which the
#       participants saw a scenario. "1" means that this particular run was the
#       1st run for that participant, "2" the second, etc.
# "Scenario"           : ID of the scenario, indicating Aid factor and Environment
# "Map"                  : The map used for that run: a large map ('M') or a small
#       map ('m')
# "UAV"                  : The size of the UAV swarms: The large swarm ('N') had 8
#       UAVs, the small one ('n') had 4
# "Environment"       : This is a convolution of 'Map' and 'UAV', i.e. a factor
#       with four levels, and as such technically redundant information
#       1: 'M'&'N' (large map, large swarm)
#       2: 'M'&'n' (large map, small swarm)
#       3: 'm'&'N' (small map, large swarm)
#       4: 'm'&'n' (small map, small swarm)
# "Aid"                  : Whether the Aid was present in a particular run ('A') or
```

```
#       not present ('a)
# "Time"                 : The completion time in seconds for that run
# "TLX.Mental"         : TLX score (0-10) for the mental workload
# "TLX.Physical"      : TLX score (0-10) for the physical workload
# "TLX.Temporal"      : TLX score (0-10) for the temporal pressure
# "TLX.Performance" : TLX score (10-0) for performance
# "TLX.Effort"         : TLX score (0-10) for effort
# "TLX.Frustration" : TLX score (0-10) for
hssData <- read.csv("./aggregated_questionnaire_data.csv", header=T)

# Print the imported lists as a sanity check
names(hssData)

# Set not automatically detected factors and create descriptive labels.
hssData$Sequence = as.factor(hssData$Sequence)
levels(hssData$Sequence) = c("1st","2nd","3rd","4th")
hssData$Environment = as.factor(hssData$Environment)

levels(hssData$Environment) = c("1 (M, N)","2 (M, n)","3 (m, N)","4 (m, n)")
levels(hssData$Aid)         = c("a", "A")
levels(hssData$Map)         = c("m","M")
levels(hssData$UAV)         = c("n","N")


## 1: Repeated measures for (completion) Time(Aid,Environment) ---------

# (no fixed effects)
# checking for intercept
m1.1<-lme(Time~1, random=~1|Participant/Aid, data=hssData, method="ML")
anova(m1.1)

# (with fixed effects)
m1.2.1<-lme(Time~Aid*Environment, random=~1|Participant/Aid/Environment, data=hssData, method="ML")
anova(m1.2.1)

m1.2.2<-lme(Time~Aid+Environment, random=~1|Participant/Aid/Environment, data=hssData, method="ML")
anova(m1.2.2)

# comparing 2.x models:
# The interaction between Aid and Environment doesn't seem to matter
```

```
anova(m1.2.1,m1.2.2)


## 2: Repeated measures for (completion) Time(Aid,Map,UAV) -------------

# Breaking up the Environment factor into UAV and Map

m2.1.1<-lme(Time~Aid*UAV*Map, random=~1|Participant/Aid/↓
→    UAV/Map, data=hssData, method="ML")
anova(m2.1.1)

m2.1.2<-lme(Time~Aid+UAV*Map, random=~1|Participant/Aid/↓
→    UAV/Map, data=hssData, method="ML")
anova(m2.1.2)

anova(m2.1.1,m1.2.1) # These two models should be identical
anova(m2.1.2,m1.2.2) # These two models should be identical

# Checking for interactions of the Aid with Map or UAV...

m2.2.1<-lme(Time~UAV+Aid*Map, random=~1|Participant/Aid/↓
→    UAV/Map, data=hssData, method="ML")
anova(m2.2.1)

m2.2.2<-lme(Time~Map+Aid*UAV, random=~1|Participant/Aid/↓
→    UAV/Map, data=hssData, method="ML")
anova(m2.2.2)

m2.2.3<-lme(Time~Aid+UAV*Map, random=~1|Participant/Aid/↓
→    UAV/Map, data=hssData, method="ML")
anova(m2.2.3)

# ... and comparing against a model withough interaction
m2.2.4<-lme(Time~Aid+UAV+Map, random=~1|Participant/Aid/↓
→    UAV/Map, data=hssData, method="ML")
anova(m2.2.4)


## 3: Repeated measures for (completion) Time(Aid,Map|UAV)  ------------

m3.1.1<-lme(Time~Aid*Map, random=~1|Participant/Aid/Map,↓
→     data=hssData, method="ML")
anova(m3.1.1)
```

```
m3.1.2<-lme(Time~Aid+Map, random=~1|Participant/Aid/Map,↓
→     data=hssData, method="ML")
anova(m3.1.2)


m3.2.1<-lme(Time~Aid*UAV, random=~1|Participant/Aid/UAV,↓
→     data=hssData, method="ML")
anova(m3.2.1)

# This seems to be the smallest model describing the data suficiently well
m3.2.2<-lme(Time~Aid+UAV, random=~1|Participant/Aid/UAV,↓
→     data=hssData, method="ML")
anova(m3.2.2)


## 4: Repeated measures for (completion) Time(Aid|Map|UAV)  ------------

m4.1<-lme(Time~Map, random=~1|Participant/Map, data=↓
→    hssData, method="ML")
anova(m4.1)

m4.2<-lme(Time~UAV, random=~1|Participant/UAV, data=↓
→    hssData, method="ML")
anova(m4.2)

m4.3<-lme(Time~Aid, random=~1|Participant/Aid, data=↓
→    hssData, method="ML")
anova(m4.3)


## 5: Details of the smallest descriptive model from 1-4 above --------------

# Checking the fixed effects section: the values for 'AidPresent' and
# 'UAVLarge Swarm' are both negative, i.e. having the aid and having more UAVs
# reduces the completion time.
summary(m3.2.2)


## 6: Participant performance in 1st scenario -------------------------------
#
# ANOVA comparing only the first run (i.e. only Sequence=="1st") data (i.e.
# all environments (M and m) and all numbers of UAVs (N and n)) to see if the
# ones using the aid (A) are indeed faster than the ones not using the aid (a).
#
# The hypotheses would be that Time(A,1st) < Time(a,1st).
```

```
dSub_1st = subset(hssData, Sequence == "1st")

m5.1<-lme(Time~1, random=~1|Participant/Aid, data=dSub_1st, method="ML")
anova(m5.1)
m5.2<-lme(Time~Aid+Map*UAV, random=~1|Participant/Aid/Map/UAV, data=dSub_1st, method="ML")
anova(m5.2)
m5.3<-lme(Time~Aid+Map+UAV, random=~1|Participant/Aid/Map/UAV, data=dSub_1st, method="ML")
anova(m5.3)
m5.4<-lme(Time~Aid+Map, random=~1|Participant/Aid/Map, data=dSub_1st, method="ML")
anova(m5.4)
m5.5<-lme(Time~Aid+UAV, random=~1|Participant/Aid/UAV, data=dSub_1st, method="ML")
anova(m5.5)
m5.6<-lme(Time~Aid, random=~1|Participant/Aid, data=dSub_1st, method="ML")
anova(m5.6)

summary(m5.6)

## Result:
# Hypothesis Time(A,1st) < Time(a,1st) is correct at alpha=0.05 (p=0.0425)


## 7: Participant improvement from 1st to 2nd scenario ----------------------
#
# ANOVA comparing the time improvement between the 1st and the 2nd run for the
# participants using the aid  (A) versus the ones not using the aid (a).
#
# The hypotheses would be that the participants _not_ using the Aid (a) improved
# more from the 1st to the 2nd run than the participants using the Aid (A), i.e.
# Time(a,2nd)-Time(a,1st) < Time(A,2nd)-Time(A,1st)
# (An improvement is a negative Delta-Time, i.e. a "smaller" improvement is
# better)

dSub_2nd = subset(hssData, Sequence == "2nd")

o1 = dSub_1st[order(dSub_1st$Participant), ] # orderd set from 1st scenario
o2 = dSub_2nd[order(dSub_2nd$Participant), ] # orderd set from 2nd scenario
```

```
185
    time_improvement = data.frame(o1$Participant,o1$Aid,o2$↓
→       Time-o1$Time)
    names(time_improvement) = c("Participant", "Aid", "↓
→       DeltaTime")

    t.test(subset(time_improvement,Aid=="a")$DeltaTime,↓
→       subset(time_improvement,Aid=="A")$DeltaTime, ↓
→       alternative="less")
190

    ## Result:
    # Hypothesis Time(a,2nd)-Time(a,1st) < Time(A,2nd)-Time(A,1st) is NOT correct
    # at alpha=0.05 (p=0.0605)

195 # The paired T-test is consevative. An alternative is to not look at the
    # improvement times, but to compare the 1st and 2nd runs with and without the
    # aid.
    # The hypthesis is that for the group not using the aid there is a difference in
    # completion time, wheras for the group having the aid there isn't.
200 # Time(a,2nd)!=Time(a,1st) and Time(a,2nd)==Time(a,1st)

    # dSub_aidPresent     = subset(hssData, Aid == "Present")
    # dSub_aidNotPresent = subset(hssData, Aid == "Not Present")
    dSub_aidPresent    = subset(hssData, Aid == "A")
205 dSub_aidNotPresent = subset(hssData, Aid == "a")

    m6.1 = lme(Time~Sequence, random=~1|Participant/Sequence↓
→       , data=subset(dSub_aidPresent, subset = Sequence %in%↓
→        c("1st","2nd")), method="ML")
    anova(m6.1)
    m6.2 = lme(Time~Sequence, random=~1|Participant/Sequence↓
→       , data=subset(dSub_aidNotPresent, subset = Sequence %↓
→       in% c("1st","2nd")), method="ML")
210 anova(m6.2)

    ## Result:
    # Hypthesis Time(a,2nd)!=Time(a,1st) and Time(a,2nd)==Time(a,1st) both are
    # true at alpha=0.05.
215


    ## 8: Maintaining performance through the four scenarios --------------------
    #
    # ANOVA checking if the expected value for completion time depends on Sequence
220 # and/or Aid presence.
    # The hypothesis is that the expected value for aided runs (A) is independent
```

```
# of the sequence ("1st","2nd","3rd","4th"), whereas the expected value for
# unaided runs (a) depends on the sequence. I.e.
# Time(A,1st)==Time(A,2nd)==Time(A,3rd)=Time(A,4th)
# versus
# Time(a,1st)!=Time(a,2nd)!=Time(a,3rd)!=Time(a,4th)

## NOTE:
# I am not sure if ANOVA can be used here as the participants change:
# There are two participant groups: G1 did a {a,a,A,A} sequence, group G2 did
# a {A,A,a,a} sequence. As such, when creating a subset on the Aid factor, the
# groups are being mixed:
# subset on 'a' -> {G1,G1,G2,G2}
# subset on 'A' -> {G2,G2,G1,G1}

m7.1.1 = lme(Time~1, random=~1|Participant/Sequence, ↓
→    data=dSub_aidPresent, method="ML")
anova(m7.1.1)
m7.1.2 = lme(Time~Sequence*Map*UAV, random=~1|↓
→    Participant/Sequence/UAV/Map, data=dSub_aidPresent, ↓
→    method="ML")
anova(m7.1.2)
m7.1.3 = lme(Time~Sequence*Map, random=~1|Participant/↓
→    Sequence/Map, data=dSub_aidPresent, method="ML")
anova(m7.1.3)
m7.1.4 = lme(Time~Sequence*UAV, random=~1|Participant/↓
→    Sequence/UAV, data=dSub_aidPresent, method="ML")
anova(m7.1.4)
m7.1.5 = lme(Time~Sequence+UAV, random=~1|Participant/↓
→    Sequence/UAV, data=dSub_aidPresent, method="ML")
anova(m7.1.5)

m7.2.1 = lme(Time~1, random=~1|Participant/Sequence, ↓
→    data=dSub_aidNotPresent, method="ML")
anova(m7.2.1)
m7.2.2 = lme(Time~Sequence*Map*UAV, random=~1|↓
→    Participant/Sequence/UAV/Map, data=dSub_aidNotPresent↓
→    , method="ML")
anova(m7.2.2)
m7.2.3 = lme(Time~Sequence*Map, random=~1|Participant/↓
→    Sequence/Map, data=dSub_aidNotPresent, method="ML")
anova(m7.2.3)
m7.2.4 = lme(Time~Sequence*UAV, random=~1|Participant/↓
→    Sequence/UAV, data=dSub_aidNotPresent, method="ML")
anova(m7.2.4)
```

```
m7.2.5 = lme(Time~Sequence, random=~1|Participant/↓
→    Sequence, data=dSub_aidNotPresent, method="ML")
anova(m7.2.5)


## Results:
# Hypothesis Time(A,1st)==Time(A,2nd)==Time(A,3rd)=Time(A,4th) is NOT correct at
# alpha=0.05 (p=0.0257).
# Hypothesis Time(a,1st)!=Time(a,2nd)!=Time(a,3rd)!=Time(a,4th) is correct at
# alpha=0.05 (p=0.006).
# I.e. the sequence always seems to matter, although it seems to matter much
# more in the unaided case (aid not present, a).
```

E.2 R Output

The relevant output from Listing (E.1) when processed in RStudio.[1]

Listing E.2: Results of the ANOVA Analysis in R.

```
> ## 1: Repeated measures for (completion) Time(Aid,↓
→    Environment) ------------
> anova(m1.1)
             numDF denDF F-value p-value
(Intercept)      1    50 995.512  <.0001


> anova(m1.2.1)
                 numDF denDF   F-value p-value
(Intercept)          1    44 1023.4138  <.0001
Aid                  1    24    6.5898  0.0169
Environment          3    44    3.9788  0.0136
Aid:Environment      3    44    1.6460  0.1925


> anova(m1.2.2)
             numDF denDF   F-value p-value
(Intercept)      1    47 1064.5017  <.0001
Aid              1    24    6.3354  0.0189
Environment      3    47    3.8915  0.0145

```

[1] Version 0.98.1091, available from `http://www.rstudio.com/`.

```
> anova(m1.2.1,m1.2.2)
       Model df      AIC      BIC    logLik   Test  L.↓
→           Ratio p-value
m1.2.1     1 12 1179.482 1210.744 -577.7412
m1.2.2     2  9 1178.660 1202.107 -580.3301 1 vs 2 ↓
→    5.177922  0.1592

> ## 2: Repeated measures for (completion) Time(Aid,Map,↓
→    UAV) ----------------
> anova(m2.1.1)
             numDF denDF   F-value p-value
(Intercept)      1    32 1023.4005  <.0001
Aid              1    24    6.5898  0.0169
UAV              1    32    3.9571  0.0553
Map              1    12    1.2146  0.2920
Aid:UAV          1    32    0.2776  0.6019
Aid:Map          1    12    3.8905  0.0720
UAV:Map          1    12    6.7647  0.0232
Aid:UAV:Map      1    12    0.7699  0.3975

> anova(m2.1.2)
             numDF denDF   F-value p-value
(Intercept)      1    33 1064.4897  <.0001
Aid              1    24    6.3354  0.0189
UAV              1    33    4.1119  0.0507
Map              1    14    1.2230  0.2874
UAV:Map          1    14    6.3397  0.0246

> anova(m2.1.1,m1.2.1) # These two models should be ↓
→    identical
       Model df      AIC      BIC    logLik   Test      ↓
→          L.Ratio p-value
m2.1.1     1 13 1181.482 1215.350 -577.7412
m1.2.1     2 12 1179.482 1210.744 -577.7412 1 vs 2 ↓
→    6.702226e-08  0.9998

> anova(m2.1.2,m1.2.2) # These two models should be ↓
→    identical
       Model df     AIC      BIC    logLik   Test      L↓
→          .Ratio p-value
m2.1.2     1 10 1180.66 1206.712 -580.3301
m1.2.2     2  9 1178.66 1202.107 -580.3301 1 vs 2 ↓
→    5.405423e-08  0.9998

> anova(m2.2.1)
```

```
              numDF denDF   F-value p-value
(Intercept)       1    33 1183.3076  <.0001
UAV               1    33    4.5926  0.0396
Aid               1    24    5.9194  0.0228
Map               1    14    1.2741  0.2780
Aid:Map           1    14    3.4947  0.0826

> anova(m2.2.2)
              numDF denDF   F-value p-value
(Intercept)       1    32 1187.4370  <.0001
Map               1    15    1.2573  0.2798
Aid               1    24    5.6551  0.0257
UAV               1    32    4.6222  0.0392
Aid:UAV           1    32    0.2382  0.6288

> anova(m2.2.3)
              numDF denDF   F-value p-value
(Intercept)       1    33 1064.4897  <.0001
Aid               1    24    6.3354  0.0189
UAV               1    33    4.1119  0.0507
Map               1    14    1.2230  0.2874
UAV:Map           1    14    6.3397  0.0246

> anova(m2.2.4)
              numDF denDF   F-value p-value
(Intercept)       1    33 1200.2526  <.0001
Aid               1    24    5.6950  0.0252
UAV               1    33    4.6733  0.0380
Map               1    15    1.2694  0.2776

> ## 3: Repeated measures for (completion) Time(Aid,Map|↓
→    UAV)  ---------------
> anova(m3.1.1)
              numDF denDF  F-value p-value
(Intercept)       1    30 992.8062  <.0001
Aid               1    24   6.0037  0.0219
Map               1    30   1.1480  0.2925
Aid:Map           1    30   3.5445  0.0695

> anova(m3.1.2)
              numDF denDF  F-value p-value
(Intercept)       1    31 997.7215  <.0001
Aid               1    24   5.7634  0.0245
Map               1    31   1.1242  0.2972

```

```
> anova(m3.2.1)
             numDF denDF   F-value p-value
(Intercept)      1    32 1156.5856  <.0001
Aid              1    24    5.6852  0.0254
UAV              1    32    4.4932  0.0419
Aid:UAV          1    32    0.2395  0.6279

> anova(m3.2.2)
             numDF denDF   F-value p-value
(Intercept)      1    33 1168.9061  <.0001
Aid              1    24    5.7250  0.0249
UAV              1    33    4.5421  0.0406

> ## 4: Repeated measures for (completion) Time(Aid|Map|↓
→     UAV)  ---------------
> anova(m4.1)
             numDF denDF   F-value p-value
(Intercept)      1    59 1001.5208  <.0001
Map              1    15    1.0659  0.3182

> anova(m4.2)
             numDF denDF   F-value p-value
(Intercept)      1    58 1271.2115  <.0001
UAV              1    16    4.6046  0.0476

> anova(m4.3)
             numDF denDF  F-value p-value
(Intercept)      1    50 985.4576  <.0001
Aid              1    24   5.8054   0.024

> ## 5: Details of the smallest descriptive model from ↓
→     1-4 above ------------
> summary(m3.2.2)
Linear mixed-effects model fit by maximum likelihood
 Data: hssData
       AIC      BIC    logLik
  1182.003 1200.239 -584.0015

Random effects:
 Formula: ~1 | Participant
        (Intercept)
StdDev:    29.70015

 Formula: ~1 | Aid %in% Participant
        (Intercept)
```

```
StdDev: 0.003302645

 Formula: ~1 | UAV %in% Aid %in% Participant
        (Intercept) Residual
StdDev:  0.01348794 78.61997

Fixed effects: Time ~ Aid + UAV
               Value Std.Error DF   t-value p-value
(Intercept) 379.2575  15.35996 33 24.691301  0.0000
AidA        -38.2000  15.96530 24 -2.392689  0.0249
UAVN        -36.1950  16.98327 33 -2.131212  0.0406
 Correlation:
     (Intr) AidA
AidA -0.520
UAVN -0.553  0.000

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.4959507 -0.6049385 -0.0939684  0.3279756  4.8671558

Number of Observations: 100
Number of Groups:
                  Participant          Aid %in% ↓
→                    Participant
                           25 ↓
→                                                 ↓
→                                50
UAV %in% Aid %in% Participant
                           84

> ## 6: Participant performance in 1st scenario ↓
→    ------------------------------
> anova(m5.1)
            numDF denDF  F-value p-value
(Intercept)     1    25 262.7958  <.0001

> anova(m5.2)
            numDF denDF  F-value p-value
(Intercept)     1    20 359.7371  <.0001
Aid             1    20   5.4889  0.0296
Map             1    20   0.4409  0.5143
UAV             1    20   1.6994  0.2072
Map:UAV         1    20   5.2241  0.0333

> anova(m5.3)
```

```
             numDF denDF   F-value p-value
(Intercept)      1    21 299.49451  <.0001
Aid              1    21   4.56968  0.0445
Map              1    21   0.36707  0.5511
UAV              1    21   1.41479  0.2475

> anova(m5.4)
             numDF denDF   F-value p-value
(Intercept)      1    22 293.95234  <.0001
Aid              1    22   4.48511  0.0457
Map              1    22   0.36028  0.5545

> anova(m5.5)
             numDF denDF   F-value p-value
(Intercept)      1    22 309.11694  <.0001
Aid              1    22   4.71649  0.0409
UAV              1    22   1.51382  0.2316

> anova(m5.6)
             numDF denDF   F-value p-value
(Intercept)      1    23 302.36220  <.0001
Aid              1    23   4.61343  0.0425

> summary(m5.6)
Linear mixed-effects model fit by maximum likelihood
 Data: dSub_1st
       AIC      BIC    logLik
  316.5202 322.6146 -153.2601

Random effects:
 Formula: ~1 | Participant
        (Intercept)
StdDev:    76.01398

 Formula: ~1 | Aid %in% Participant
        (Intercept) Residual
StdDev:    76.01398 28.50524

Fixed effects: Time ~ Aid
               Value Std.Error DF  t-value p-value
(Intercept) 455.0833  33.47181 23 13.59602  0.0000
AidA        -99.6987  46.41705 23 -2.14789  0.0425
 Correlation:
     (Intr)
AidA -0.721
```

```
Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         ↓
→           Max
-0.44498152 -0.13455369 -0.04006474  0.12125787  ↓
→    0.83868551

Number of Observations: 25
Number of Groups:
         Participant Aid %in% Participant
                  25                   25

> ## 7: Participant improvement from 1st to 2nd ↓
→    scenarios ------------------

> t.test(subse .... [TRUNCATED]

  Welch Two Sample t-test

data:  subset(time_improvement, Aid == "a")$DeltaTime ↓
→    and
       subset(time_improvement, Aid == "A")$DeltaTime
t = -1.6505, df = 14.062, p-value = 0.0605
alternative hypothesis: true difference in means is less↓
→     than 0
95 percent confidence interval:
     -Inf 5.665307
sample estimates:
 mean of x  mean of y
-115.91667  -31.15385

> anova(m6.1)
            numDF denDF F-value p-value
(Intercept)     1    12 720.362  <.0001
Sequence        1    12   2.980  0.1099

> anova(m6.2)
            numDF denDF  F-value p-value
(Intercept)     1    11 242.2186  <.0001
Sequence        1    11   5.8120  0.0346

> ## 8: Maintaining performance through the four ↓
→    scenarios ------------------
> anova(m7.1.1)
            numDF denDF F-value p-value
```

```
(Intercept)     1    25  1174.5  <.0001

> anova(m7.1.2)
                 numDF denDF  F-value p-value
(Intercept)          1    24 868.8601  <.0001
Sequence             3    10   6.5239  0.0101
Map                  1    10   0.0042  0.9499
UAV                  1    10  20.6032  0.0011
Sequence:Map         3    10   5.1337  0.0210
Sequence:UAV         3    10   1.7556  0.2189
Map:UAV              1    10  11.7854  0.0064
Sequence:Map:UAV     3    10   1.4961  0.2747

> anova(m7.1.3)
             numDF denDF   F-value p-value
(Intercept)      1    24 1211.9185  <.0001
Sequence         3    18    3.0886  0.0533
Map              1    18    0.0669  0.7988
Sequence:Map     3    18    1.6906  0.2047

> anova(m7.1.4)
             numDF denDF   F-value p-value
(Intercept)      1    24 1537.9027  <.0001
Sequence         3    18    3.6044  0.0337
UAV              1    18   13.0110  0.0020
Sequence:UAV     3    18    0.1955  0.8981

> anova(m7.1.5)
            numDF denDF   F-value p-value
(Intercept)     1    24 1710.4849  <.0001
Sequence        3    21    3.7870  0.0257
UAV             1    21   13.8628  0.0013

> anova(m7.2.1)
            numDF denDF  F-value p-value
(Intercept)     1    25 504.6871  <.0001

> anova(m7.2.2)
                 numDF denDF  F-value p-value
(Intercept)          1    24 473.8586  <.0001
Sequence             3    10  12.1300  0.0011
Map                  1    10   3.5779  0.0878
UAV                  1    10   0.1990  0.6650
Sequence:Map         3    10   2.6417  0.1067
Sequence:UAV         3    10   0.2338  0.8708
```

```
Map:UAV               1    10  16.2411  0.0024
Sequence:Map:UAV      3    10   7.7540  0.0058

> anova(m7.2.3)
             numDF denDF  F-value p-value
(Intercept)      1    24 644.2027  <.0001
Sequence         3    18   5.6615  0.0065
Map              1    18   2.9080  0.1053
Sequence:Map     3    18   1.2426  0.3236

> anova(m7.2.4)
             numDF denDF  F-value p-value
(Intercept)      1    24 610.7474  <.0001
Sequence         3    18   5.0244  0.0105
UAV              1    18   1.2509  0.2781
Sequence:UAV     3    18   0.2332  0.8720

> anova(m7.2.5)
            numDF denDF  F-value p-value
(Intercept)     1    24 583.8868  <.0001
Sequence        3    22   5.4362   0.006
```

REFERENCES

[1] "Gnu compiler collection 4.9 release series.". Available via: https://gcc.gnu.org/gcc-4.9/, retrieved Dec 2014. 3.4

[2] Adams, J. A., "Unmanned vehicle situation awareness: A path forward," in *Human systems integration symposium*, pp. 31–89, 2007. 2.3, 2.4

[3] Adams, J. A., "Critical considerations for human-robot interface development," in *AAAI Fall Symposium*, pp. 1–7, 2002. 2.3, 2.4

[4] Adams, J. A., "Human-robot interaction design: Understanding user needs and requirements," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 49, pp. 447–451, SAGE Publications, 2005. 2.3

[5] Adams, J. A., Humphrey, C. M., Goodrich, M. A., Cooper, J. L., Morse, B. S., Engh, C., and Rasmussen, N., "Cognitive task analysis for developing unmanned aerial vehicle wilderness search support," *Journal of Cognitive Engieneering and Decision Making*, vol. 3, pp. 1–26, mar 2009. DOI:10.1518/155534309X431926. 2.3

[6] Ben-Moshe, B., Hall-Holt, O., Katz, M. J., and Mitchell, J. S. B., "Computing the visibility graph of points within a polygon," in *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, (New York, NY, USA), p. 27–35, ACM, 2004. DOI:10.1145/997817.997825. 4.3.2.1

[7] Berg, B., Simon, C., Schwenk, F., and SDAPS Community, "Scripts for data acquisition with paper-based surveys (SDAPS).". Available via: http://sdaps.org, retrieved mar 2015. B.3, B.4, D

[8] Brewer, C. A., "Color use guidelines for mapping and visualization," vol. 2 of *Modern cartographie*, ch. 7, pp. 123–147, New York: Pergamon, 1st ed., 1997. Available via: http://www.sciencedirect.com/science/bookseries/13630814/2. 7

[9] Brewer, C. A., Harrower, M., and The Pennsylvania State University, "Colorbrewer 2.0 (color advice for cartography).". Available via: http://colorbrewer2.org/. 7

[10] Bruin, H., "Row complete latin squares.". Available via: http://personal.maths.surrey.ac.uk/st/H.Bruin/MMath/LatinSquares.html, retrieved mar 2015. C.1

[11] Calhoun, G., Ruff, H., Draper, M., Wright, N., and Mullins, B., "Levels of automation in multi-uav control allocation and router tasks," in *AIAA Infotech@Aerospace Conference*, 2009. AIAA-2009-1947. 2.2.4

[12] Calhoun, G. L. and Draper, M. H., *Multi-Sensory Interfaces for Remotely Operated Vehicles*, ch. 11, pp. 149 – 163. Vol. 7 of *Advances in Human Performance and*

Cognitive Engineering Research [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07011-6. 2.2.5

[13] Christmann, H. C., "Self-configuring ad-hoc networks for unmanned aerial systems," Master's thesis, Georgia Institute of Technology, Atlanta, GA, may 2008. Available via: http://hdl.handle.net/1853/22626. 3.3.1

[14] Christmann, H. C., "MVS human subject study data set: Coverage aid," may 2015. Available via: http://hdl.handle.net/1853/53344. 5.3, 5.3.1, 14, 6.1

[15] Christmann, H. C., "MVS software framework," 2015. Available via: https://github.com/mvs-framework, retrieved May 2015. 6.1

[16] Christmann, H. C., Christophersen, H. B., Wu, A. D., and Johnson, E. N., "Guidance, navigation, control, and operator interfaces for small rapid response unmanned helicopters," in *AHS 64th Annual Forum and Technology Display*, (Montréal, Canada), apr 2008. Available via: http://hdl.handle.net/1853/35874. 1.2

[17] Christmann, H. C. and Johnson, E. N., "Design and implementation of a self-configuring ad-hoc network for unmanned aerial systems," in *AIAA InfoTech@AerospaceConference*, (Rohnert Park, CA, United States), may 2007. AIAA-2007-2779. DOI:10.2514/6.2007-2779. 3.3.1

[18] Christmann, H. C. and Johnson, E. N., "UAS Reference Scenarios for MANET Development," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, (Honolulu, HI, USA), aug 2008. AIAA-2008-7042. DOI:10.2514/6.2008-7042.

[19] Christmann, H. C. and Johnson, E. N., "Modeling urban environments for communication-aware uav swarm path planning," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, (Toronto, Ontario, Canada), 2010. AIAA-2010-8361. DOI:10.2514/6.2010-8361. 3.3.1

[20] Christmann, H. C., "Row-complete latin squares in reduced form.". Available via: https://gist.github.com/hcc23/274b76edc3dd82d5de1a, retrieved mar 2015. C.1

[21] Chun, W. H., Spura, T., Alvidrez, F. C., and Stiles, R. J., *Spatial Dialog and Unmanned Aerial Vehicles*, ch. 14, pp. 193 – 206. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07014-1. 2.2.5

[22] Cooke, N. J., Pedersen, H. K., Connor, O., Gorman, J. C., and Andrews, D., *Acquiring Team-Level Command and Control Skill for UAV Operation*, ch. 20, pp. 285–297. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. 2.2.2

[23] Cooper, J. and Goodrich, M. A., "Towards combining uav and sensor operator roles in uav-enabled visual search," in *3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, (Amsterdam, Netherlands), pp. 351–358, mar 2008. 2.3, 2.4

[24] Cummings, M. L. and Bruni, S., "Collaborative human-computer decision making in network centric warfare," in *TTCP HUM TP7: Workshop on Aerospace Human Factors Issues in Network-Centric Warfare. Sailsbury, UK*, 2005. 2.2.5

[25] Cummings, M. L., Bruni, S., Mercier, S., and Mitchell, P. J., "Automation architecture for single operator, multiple uav command and control," *The International C2 Journal*, vol. 1, no. 2, pp. 1–24, 2007. 1, 2.2, 2.2.4

[26] Cummings, M. L. and Guerlain, S., "Developing operator capacity estimates for supervisory control of autonomous vehicles," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 49, pp. 1–15, Feb. 2007. DOI:10.1518/001872007779598109. 2.2.3, 2.2.4, 2.3

[27] Cummings, M. L. and Mitchell, P. J., "Operator scheduling strategies in supervisory control of multiple uavs," *Aerospace Science and Technology*, vol. 11, no. 4, pp. 339 – 348, 2007. COGIS '06. DOI:DOI: 10.1016/j.ast.2006.10.007. 2.2.5

[28] Cummings, M. L. and Mitchell, P. J., "Predicting controller capacity in supervisory control of multiple uavs," *Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, pp. 451–460, Mar. 2008. 2.2.4, 2.3

[29] Cummings, M. L. and Nehme, C. E., "Modeling the impact of workload in network centric supervisory control settings.". 2.2.3

[30] Cummings, M. L., Pina, P., and Crandall, J. W., "A metric taxonomy for supervisory control of unmanned vehicles," in *Proccedings of AUVSI's Unmanned Systems North America*, 2009. 2.2.3

[31] Cummings, M. L., Bertucelli, L. F., Macbeth, J., and Surana, A., "Task versus vehicle-based control paradigms in multiple unmanned vehicle supervision by a single operator," *IEEE Transactions on Human-Machine Systems*, vol. 44, pp. 353–361, jun 2014. DOI:10.1109/THMS.2014.2304962. 2.3, 2.4

[32] Cummings, M. L., Brzezinski, A. S., and Lee, J. D., "Operator performance and intelligent aiding in unmanned aerial vehicle scheduling," *Intelligent Systems, IEEE*, vol. 22, pp. 52–59, March-April 2007. DOI:10.1109/MIS.2007.39. 2.2.3, 2.2.4

[33] Cummings, M. L. and Clare, A. S., "Holistic modelling for human-autonomous system interaction," *Theoretical Issues in Ergonomics Science*, vol. 16, pp. 214–231, mar 2015. DOI:10.1080/1463922X.2014.1003990. 2.3, 2.4

[34] de la Croix, J.-P. and Egerstedt, M., "Controllability characterizations of leader-based swarm interactions.," in *AAAI Fall Symposium: Human Control of Bioinspired Swarms*, (Arlington, DC), Nov. 2012. 2.3, 2.4

[35] DeJoode, J. A., Cooke, N. J., Shope, S. M., and Pedersen, H. K., *Guiding the Design of a Deployable UAV Operations Cell*, ch. 22, pp. 311–328. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07022-0. 2.2.1

[36] Digia Plc. and others, "Qt.". Available via: `https://www.qt.io/`, retrieved dec 2014. 3.4

[37] Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, p. 269–271, 1959. DOI:10.1007/BF01386390. 3.3.3

[38] Ding, X. C., Powers, M., Egerstedt, M., yih Young, S., and Balch, T., "Pilot decision support for controlling multiple uavs," *IEEE Robotics and Automation Magazine*, vol. 1 6, no. 2, pp. 73–81, 2009. DOI:10.1109/MRA.2009.932526. 2.3

[39] Dixon, S. R. and Wickens, C. D., "Control of multiple-UAVs: a workload analysis," in *Proceedings of the 12th International Symposium on Aviation Psychology*, 2001. 2.2.3

[40] Drury, J. L., Riek, L., and Rackliffe, N., "A decomposition of UAV-related situation awareness," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, (Salt Lake City, Utah, USA), pp. 88–94, ACM, 2006. DOI:10.1145/1121241.1121258. 2.2.2

[41] Endsley, M. R., "Situation awareness global assessment technique (sagat)," in *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, vol. 3, p. 789 –795, may 1988. DOI:10.1109/NAECON.1988.195097. (document)

[42] Endsley, M. R., "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, p. 32–61, mar 1995. T58.A2 H8. (document)

[43] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Professional Computing Series, Addison-Wesley, 1995. 25

[44] Gehrels, B., Lalande, B., Loskot, M., Wulkiewicz, A., and Karavelas, M., "Boost.Geometry," 2009–2014. Available via: `http://www.boost.org/doc/libs/1_57_0/libs/geometry/doc/html/index.html`, retrieved Jan 2015. 3

[45] Goodrich, M. A. and Cummings, M. L., *Human Factors Perspective on Next Generation Unmanned Aerial Systems*, ch. 99, pp. 2405–2423. Springer Netherlands, 2014. DOI:10.1007/978-90-481-9707-1. 2.3, 2.4

[46] Goodrich, M. A., L., C. J., Adams, J. A., Humphrey, R. Z., and Buss, B. G., "Using a mini-uav to support wilderness search and rescue: Practices for human-robot teaming," in *IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 1–6, sep 2007. DOI:10.1109/SSRR.2007.4381284. 2.3

[47] Goodrich, M. A., McLain, T. W., Anderson, J. D., Sun, J., and Crandall, J. W., "Managing autonomy in robot teams: Observations from four experiments," in *2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 25–32, mar 2007. 2.3, 2.4

[48] Goodrich, M. A. and Schultz, A. C., "Human-robot interaction: A survey," *Foundation and Trends in Human-Computer Interaction*, vol. 1, pp. 203–275, Jan. 2007. 2.2, 2.3, 2.4

[49] Guennebaud, G., Jacob, B., and others, "Eigen v3," 2010. Available via: `https://eigen.tuxfamily.org`, retrieved Dec 2014. 3.4

[50] Harel, D., "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, pp. 231–274, jun 1987. DOI:10.1016/0167-6423(87)90035-9. 25, 3.5

[51] Hawick, K. A. and James, H. A., "Enumerating circuits and loops in graphs with self-arcs and multiple-arcs," in *Proceedings of the 2008 International Conference on Foundations of Computer Science, FCS 2008, July 14-17, 2008, Las Vegas, Nevada, USA* (Arabnia, H. R., Mun, Y., and Zhou, P. L., eds.), pp. 14–20, CSREA Press, 2008. Available via: `http://www.massey.ac.nz/~kahawick/cstn/013/cstn-013.pdf`, retrieved mar 2015. 38

[52] Held, M., "VRONI An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments," *Computational Geometry*, vol. 18, p. 95–123, mar 2001. DOI:10.1016/S0925-7721(01)00003-7. 3.2.2.1, 3.4, 34

[53] Hou, M. and Kobierski, R. D., *Operational Analysis and Performance Modeling for the Control of Multiple Uninhabited Aerial Vehicles from an Airborne Platform*, ch. 19, pp. 267–282. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07019-0. 2.2.2

[54] Hou, M., Kobierski, R. D., and Brown, M., "Intelligent adaptive interfaces for the control of multiple UAVs," *Journal of Cognitive Engineering and Decision Making*, vol. 1, pp. 327–362, 2007. DOI:10.1518/155534307X255654. 2.2.2

[55] Humphrey, C. M., Henk, C., Sewell, G., Williams, B. W., and Adams, J. A., "Assessing the scalability of a multiple robot interface," in *2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, (Arlington, VA, USA), pp. 239–246, Mar. 2007. 2.3, 2.4

[56] Joint Education and Doctrine Division, J-7, Joint Staff, "Department of defense dictionary of military and associated terms," Nov. 2014. Joint Publication 1-02. Available via: `http://dtic.mil/doctrine/new_pubs/jp1_02.pdf`, retrieved apr 2015. (document)

[57] Joint Education and Doctrine Division, J-7, Joint Staff, "Command and control of joint air operations," Feb. apr 2015. Joint Publication 3-30. Available via: http://dtic.mil/doctrine/new_pubs/jp3_30.pdf. (document)

[58] Kim, J., Zhang, F., and Egerstedt, M., "An exploration strategy by constructing voronoi diagrams with provable completeness," in *Proceedings of the 48th IEEE Conference on Decision and Control*, (Shanghai), pp. 7024–7029, Dec. 2009. DOI:10.1109/CDC.2009.5400798. 3.2.2.1

[59] Ko, Y.-B. and Vaidya, N. H., "Location-aided routing (lar) in mobile ad hoc networks," in *Mobile Computing and Networking*, p. 66–75, 1998. Available via: http://sigmobile.org/awards/mobicom1998-student.pdf. 3.3.2

[60] Lantinga, S. and SDL Community, "Simple directmedia layer (sdl).". Available via: http://http://www.libsdl.org, retrieved mar 2015. 23

[61] Levinthal, B. R. and Wickens, C. D., "Management of multiple uavs with imperfect automation," *Human Factors and Ergonomics Society Annual Meeting Proceedings*, vol. 50, pp. 1941–1944, 2006. DOI:10.1177/154193120605001748. 2.2.3

[62] Lewis, M., Polvichai, J., Sycara, K., and Scerri, P., *Scaling-up Human Control for Large UAV Teams*, ch. 17, pp. 237 – 250. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07017-7. 2.2.5

[63] MathWorks, "Stateflow: Model and simulate decision logic using state machines and flow charts.". Available via: http://www.mathworks.com/products/stateflow/, retrieved feb 2015. 3.5

[64] McGill, R., Tukey, J. W., and Larsen, W. A., "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. pp. 12–16, 1978. DOI:10.2307/2683468. 8

[65] Miller, C. A., Funk, H., Dorneich, M., and Whitlow, S. D., "A playbook interface for mixed initiative control of multiple unmanned vehicle teams," in *Proceedings. The 21st Digital Avionics Systems Conference*, vol. 2, (Irvine, CA, USA), pp. 7E4–1 – 7E4–13, 2002. DOI:10.1109/DASC.2002.1052935. 2.2.5, 2.3

[66] Miller, C. A. and Parasuraman, R., "Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 49, pp. 57–75, Feb. 2007. DOI:10.1518/001872007779598037. 2.2.5, 2.3

[67] Morales, Y., Satake, S., Kanda, T., and Hagita, N., "Building a model of the environment from a route perspective for human–robot interaction," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 165–181, 2015. DOI:10.1007/s12369-014-0265-8. 2.2.5

[68] Naikar, N., Pearce, B., Drumm, D., and Sanderson, P. M., "Designing teams for first-of-a-kind, complex systems using the initial phases of cognitive work analysis: Case study," *Human Factors*, vol. 45, pp. 202–217, 2003. 2.2.2

[69] Naval Research Laboratory, "Interactive scenario builder.". Available via: https://builder.nrl.navy.mil, retrieved Jan. 2015. 3.3

[70] NS-3 Consortium, "The network simulator - ns-3.". Available via: http://www.nsnam.org/, retrieved Jan. 2015. 3.3

[71] Obermeyer, K. J. and Contributors, "The VisiLibity library," 2008. R-1. Available via: http://www.VisiLibity.org, retrieved jul 2010. 3.3.2, 3.4

[72] O'Rourke, J., *Art Gallery Theorems and Algorithms*. No. 3 in The International Series of Monographs on Computer Science, New York: Oxford University Press, 1987. Available via: http://cs.smith.edu/~orourke/books/ArtGalleryTheorems/Art_Gallery_Full_Book.pdf. 4.3.2.2

[73] Parasuraman, R. and Miller, C., *Delegation Interfaces for Human Supervision of Multiple Unmanned Vehicles: Theory, Experiments, and Practical Applications*, ch. 18, pp. 251–266. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07018-9. 2.2.5

[74] Real-Time Innovations, Inc., "Connext DDS Professional.". Available via: http://www.rti.com/products/dds/index.html, retrieved Dec. 2014. 3.1, 3.4

[75] Real-Time Innovations, Inc., "Research and Collaboration University Program.". Available via: http://www.rti.com/resources/research-programs.html#UNIVERSITY, retrieved mar 2015. 2

[76] Sack, J.-R. and Urrutia, J., eds., *Handbook of Computational Geometry*. Amsterdam: North-Holland / Elsevier, 2000. 4.3.2.2

[77] Salas, E., *Human Factors of Remotely Operated Vehicles*, vol. 7 of *Advances in Human Performance and Cognitive Engineering Research*. Emerald Group Publishing Limited, 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. 1, 2.2, 2.2.5, E.2

[78] Sheridan, T. B., *Telerobotics, Automation, and Human Supervisory Control*. The MIT Press, 1992. Available via: https://mitpress.mit.edu/books/telerobotics-automation-and-human-supervisory-control. (document), 2.2

[79] Sheridan, T. B. and Verplank, W. L., "Human and computer control of undersea teleoperators," Tech. Rep. ADA057655, MIT Man-Machine Systems Lab, Cambridge, MA, March 1978. Available via: http://handle.dtic.mil/100.2/ADA057655. (document), 2.2.4

[80] Shope, S. M., DeJoode, J. A., Cooke, N. J., and Pedersen, H., "Using pathfinder to generate communication networks in a cognitive task analysis," *Human Factors and Ergonomics Society Annual Meeting Proceedings*, vol. 48, pp. 678–682, 2004. DOI:10.1177/154193120404800386. 2.2.1, 2.2.5

[81] Siek, J., Lee, L.-Q., Lumsdaine, A., Sutton, A., and Willcock, J., "Boost.Graph." http://www.boost.org, 2000–2001. Available via: `https://www.boost.org/doc/libs/1_57_0/libs/graph/`. 3.4, 38

[82] Simons, D. J. and Chabris, C. F., "Gorillas in our midst: Sustained inattentional blindness for dynamic events," *Perception*, vol. 28, p. 1059–1074, 1999. DOI:10.1068/p2952. 1.2

[83] Solovey, E., Jackson, K., and Cummings, M. L., "Collision avoidance interface for safe piloting of unmanned vehicles using a mobile device," in *25th annual ACM symposium on User Interface software and Technology*, (Cambridge, MA, USA), pp. 77–78, nov 2012. DOI:10.1145/2380296.2380330. 2.3, 2.4

[84] The KDE Project. Available via: `https://techbase.kde.org/Policies/Library_Code_Policy#D-Pointers`, retrieved Jan 2015. 3.4.1

[85] The Qt Company Ltd., "Qt 5.4: Qt Licensing.". Available via: `https://doc.qt.io/qt-5/licensing.html`, retrieved feb 2015. 4.1

[86] The Qt Company Ltd., "Qt 5.4: Supported Platforms.". Available via: `https://doc.qt.io/qt-5/supported-platforms.html`, retrieved feb 2015. 4.1

[87] The Qt Project. Available via: `http://qt-project.org/wiki/Dpointer`, retrieved Jan 2015. 3.4.1

[88] Tompkins, A., "Boost.Uuid." `http://www.boost.org`, 2006. Available via: `https://www.boost.org/doc/libs/1_57_0/libs/uuid/`, retrieved Dec 2014. 3.4

[89] Tso, K., Tharp, G., Tai, A., Draper, M., Calhoun, G., and Ruff, H., "A human factors testbed for command and control of unmanned air vehicles," in *Digital Avionics Systems Conference, 2003. DASC '03. The 22nd*, vol. 2, pp. 8.C.1–81–12, 2003. DOI:10.1109/DASC.2003.1245899. 2.2.4

[90] Wanless, I. M., "Data on complete and row-complete latin squares.". Available via: `http://users.monash.edu.au/~iwanless/data/RCLS/index.html`, retrieved mar 2015. C.1

[91] Weil, S. A., Freeman, J., MacMillan, J., Jackson, C. D., Mauer, E., Patterson, M. J., and Linegang, M. P., *Design of a Multi-Vehicle Control System: System Design and User Interaction*, ch. 16, pp. 223–236. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07016-5. 2.2.5

[92] Wickens, C. D., Dixon, S. R., and Ambinder, M. S., *Workload and Automation Reliability in Unmanned Air Vehicles*, ch. 15, pp. 209–222. Vol. 7 of *Advances in Human Performance and Cognitive Engineering Research* [77], 1 ed., 2006. Editors: Nancy J. Cooke, Heather L. Pringle, Harry K. Pedersen, Olena Connor. DOI:10.1016/S1479-3601(05)07015-3. 2.2.3

[93] Wickham, H., "ggplot2 Help Topics.". Available via: `http://docs.ggplot2.org`, retrieved mar 2015. 8

[94] Wikipedia, "Osi model.". Available via: `https://en.wikipedia.org/wiki/OSI_model`, retrieved Jan. 2015. (document)

[95] Yerkes, R. M. and Dodson, J. D., "The relation of strength of stimulus to rapidity of habit-formation," *Journal of Comparative Neurology and Psychology*, vol. 18, pp. 459–482, 1908. 2.2.3